НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

Магістерська дисертація

зі спеціальності 121 Інженерія програмного забезпечення
за спеціалізацією Інженерія програмного забезпечення розподілених систем
на тему Підсистема інтелектуального редактора природномовних текстів

Виконав: студент 6 курсу, групи ТВ-82мп
_____Гольдич Ярослав Євгенович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц. к.т.н. доц. Стативка Ю.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Інженерія програмного забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)

«____» _____ 2019р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Гольдичу Ярославу Євгеновичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Підсистема інтелектуального редактора природномовних текстів

Науковий керівник Стативка Юрій Іванович доц. к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “04 ” листопада 2019 року № 3812-с

2. Строк подання студентом дисертації 09.12.2019

3. Об’єкт дослідження Підсистема інтелектуального редактора природномовних текстів

4. Предмет дослідження Підсистема інтелектуального редактора природномовних текстів на основі Visual Studio Code

5. Перелік питань, які потрібно розробити
   1) розглянути особливості природних мов та їх інструментальне забезпечення;
   2) розробити архітектуру розширюваного середовища для написання природномовних текстів;
   3) побудувати програмну реалізацію системи;
   4) описати та розробити протокол мовного сервера;
   5) розробити опис використання програмної системи;
   6) розробити інтерфейс користувача.

Орієнтований перелік ілюстративного матеріалу
   1) особливості природних мов;

2) архітектура мовного клієнта;

3) архітектура мовного сервера;

4) особливості протокола мовного сервера;

5) сценарій використання редактора природомовних текстів.

6. Орієнтований перелік публікацій
XVII-а міжнародна науково-практична конференція аспірантів, магістрантів, студентів "Сучасні проблеми наукового забезпечення енергетики"

7. Дата видачі завдання «12» вересня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1.	Затвердження теми роботи		виконано
2.	Вивчення та аналіз задачі	1.01-1.02.2019	виконано
3.	Розробка архітектури та загальної структури системи	2.02-15.03.2019	виконано
4.	Розробка структур окремих підсистем	16.03-1.05.2019	виконано
5.	Програмна реалізація системи	2.05-21.10.2019	виконано
6.	Захист програмного продукту	22.10.2019	виконано
7.	Оформлення пояснювальної записки	23.10-20.11.2019	виконано
8.	Передзахист	21.11.2019	виконано
9.	Захист		

Студент _____ Гольдич Я. Є.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Стативка Ю. І
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Магістерська дисертація складається зі вступу, чотирьох розділів, висновку, переліку посилань з 30 найменувань, 1 додатку, і містить 20 рисунків, 31 таблицю. Повний обсяг магістерської дисертації складає 82 сторінки, з яких перелік посилань займає 3 сторінки, додатки – 3 сторінки.

Актуальність теми. Проблема асистування створенню природномовних текстів здебільшого зводиться лише до перевірки — орфографії та синтаксису, а підказки рідко виходять за межі доповнення префіксу до леми чи найчастотнішої словоформи, саме тому тема роботи є актуальною, вихідний продукт роботи підвищить ефективність та якість написання природномовних текстів.

Зв’язок роботи з науковими програмами, планами, темами. Робота відноситься до наукової програми «Методи та засоби підвищення ефективності природномовних програмних систем».

Мета дослідження — побудова підсистеми інтелектуального редактора природномовних текстів та створення мовного сервера.

Для досягнення поставленої задачі були сформульовані наступні **завдання дослідження**, що визначили логіку дослідження та його структуру: розглянути особливості природномовних текстів та інструментальне забезпечення для їх використання, розробити та реалізувати архітектуру розширюваного середовища для написання природномовних текстів, розробити архітектуру мовного сервера для написання природномовних текстів.

Об’єкт дослідження — підсистема інтелектуального редактора природномовних текстів.

Предмет дослідження — підсистема інтелектуального редактора природномовних текстів на основі Visual Studio Code.

Наукова новизна одержаних результатів. В процесі виконання роботи набуло подальшого розвитку надання мовної підтримки для написання текстів природною мовою за рахунок: створення синтаксичних аналізаторів з врахуванням афіксів, удосконалення провайдерів розумного доповнення, створення мовного сервера, удосконалення протоколу мовного сервера.

Створення синтаксичних аналізаторів з врахуванням афіксів забезпечує більш точне обчислення релевантності лексеми шляхом врахування контексту. Удосконалення провайдерів розумного доповнення забезпечує зменшення витраченого часу при написанні групи лексем.

Створення виділеного мовного сервера забезпечує зменшення навантаження на мовний клієнт, можливості оновлення мовних провайдерів без необхідності оновлення мовного клієнта та покращило швидкість надання мовної підтримки.

Протокол мовного сервера удосконалено за рахунок обрання протоколу більш низького рівня в якості транспортного, це забезпечує зменшення обсягу повідомлення.

Апробація результатів дисертації. Результати досліджень оприлюднені на XVII-й міжнародній науково-практичній конференції аспірантів, магістрантів, студентів "Сучасні проблеми наукового забезпечення енергетики".

Публікації. Результати дисертації опубліковані у матеріалах XVII-й міжнародній науково-практичній конференції аспірантів, магістрантів, студентів "Сучасні проблеми наукового забезпечення енергетики".

Ключові слова: *ПРИРОДНОМОВНИЙ ТЕКСТ, МОВНИЙ КЛІЄНТ, МОВНИЙ СЕРВЕР, VISUAL STUDIO CODE, РОЗШИРЕННЯ, АНАЛІЗАТОР*.

ABSTRACT

Structure and scope of the thesis. The master's thesis consists of an introduction, four sections, conclusion, a list of references of 30 titles, 1 appendix, and contains 20 figures, 31 tables. The full volume of the master's thesis is 82 pages, of which the list of links occupies 3 pages, the appendices - 3 pages.

Actuality of theme. The problem of assisting with the creation of natural-language texts is mostly confined to spelling and syntax, and clues rarely go beyond adding a prefix to a lemma or the most frequent word-form, which is why the topic of work is relevant, the original product of the work will increase the efficiency and quality of writing natural-language texts.

Relationship with working with scientific programs, plans, topics. The work is related to the scientific program "Methods and Means of Improving the Efficiency of Natural-Language Software Systems".

The purpose of the study is to build a subsystem of intelligent editor of natural language texts and to create a language server.

To achieve this task, the following **research objectives** were formulated, which defined the logic of the study and its structure: to consider the features of natural-language texts and tools for their use, to develop and implement an architecture of extensible environment for writing natural-language texts, to develop a language server architecture for writing natural-language texts.

The object of study is the subsystem of intelligent editor of natural-language texts.

The subject of the research is the subsystem of intelligent editor of natural-language texts on the basis of Visual Studio Code.

Scientific novelty of the obtained results. In the course of the work, the development of language support for writing texts in natural language was further developed through the creation of affix parser, perfection of smart supplement providers, creation of the language server, and improvement of the language server protocol.

Creating affixed parsers provides more accurate computation of the token's relevance by context. Improving Reasonable Provisioning Providers reduces the time spent writing a token group.

Creating a dedicated language server reduces the load on the language client, upgrades the language providers without the need to update the language client, and improves the speed of language support.

The language server protocol has been improved by choosing a lower level protocol as the transport protocol, which reduces the volume of the message.

Testing the results of the thesis. The results of the research were published at the XVII-th International Scientific and Practical Conference of postgraduate students, undergraduates, students "Modern problems of scientific support of energy".

Publications. The results of the dissertation are published in the materials of the XVII-th international scientific-practical conference of postgraduate students, undergraduates, students "Modern problems of scientific support of energy".

Keywords: *NATURAL LANGUAGE TEXT, LANGUAGE CLIENT, LANGUAGE SERVER, VISUAL STUDIO CODE, EXTENSION, PARSER*.

ЗМІСТ

Вступ 10
1. Засоби для написання природномовних текстів 12
1.1. Особливості написання природномовних текстів 13
1.2. Інструменти для написання природномовних текстів 18
1.3. Написання природномовних текстів у Visual Studio Code 20
2. Архітектура інструменту написання природномовних текстів 22
2.1. Базові засоби розширення Visual Studio Code 22
2.1.1. Сервер мови 25
2.1.3. Маніфест розширення 27
2.1.4. Особливості API Visual Studio Code 29
2.1.5. Публікація розширення 30
2.2. Архітектура середовища написання природномовних текстів 31
3. Програмна реалізація 33
3.1. Мовний клієнт 35
3.1.1 Модуль діагностики 37
3.1.2 Модуль розумного доповнення 38
3.1.3 Модуль підказки 39
3.1.4 Людино-машинна взаємодія 41
3.2. Мовний сервер 43
3.2.1. Протокол мовного сервера 46
3.2.2. Лексичний аналізатор 50
3.2.3. Виконання запитів мовного клієнта 51
3.2.4. Забезпечення підтримки природномовних текстів 52
4. Розроблення стартап-проекту 59
4.1. Опис ідеї проекту 59
4.2. Технологічний аудит ідеї проекту 61
4.3. Аналіз ринкових можливостей запуску стартап-проекту 62

4.4. Базова стратегія розвитку 67
4.5. Розроблення маркетингової програми стартап-проекту 69
Висновки 75
Список використаних джерел 76
Додаток А 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VS Code — Visual Studio Code.

LSP — Language server protocol, протокол мовного сервера.

TCP — Transmission Control Protocol, протокол керування передачею.

DER — Distinguished Encoding Rules

JSON — JavaScript Object Notation, запис об’єктів JavaScript.

API — Application Programming Interface, інтерфейс прикладного програмування.

ASN — Abstract Syntax Notation.

COM — модель об'єктних компонентів.

RPC — Remote Procedure Call, виклик віддалених процедур.

UDP — User Datagram Protocol, протокол датаграм користувача.

ВСТУП

Така функція редактора текстів, як інтелектуальне асистування, допомагає значно скоротити час написання, рецензування та редагування текстів.

На даний момент майже для кожної штучної мови існує велика кількість редакторів чи повнофункціональних середовищ розробки, оскільки для штучної мови заздалегідь зазначені граматика, алфавіт та словник, вони є повністю формальними мовами (мають скінченну послідовність символів, які описуються правилами певного виду).

Разом з тим проблема асистування створенню природномовних текстів здебільшого зводиться лише до перевірки — орфографії, синтаксису та, можливо, певних стилістичних показників тексту, а підказки рідко виходять за межі доповнення префіксу до леми чи найчастотнішої словоформи.

Необхідність розширення репертуару перевірок та підказок для підвищення ефективності роботи користувача, який створює чи редагує природномовний текст і визначає актуальність обраної теми.

Об’єкт дослідження — підсистема інтелектуального редактора природномовних текстів.

Предмет дослідження — підсистема інтелектуального редактора природномовних текстів на основі Visual Studio Code.

Мета дослідження — побудова підсистеми інтелектуального редактора природномовних текстів на основі Visual Studio Code.

Для досягнення поставленої мети необхідно виконати такі завдання:

- розглянути особливості природномовних текстів та інструментальне забезпечення для їх використання;
- розробити архітектуру розширюваного середовища для написання природномовних текстів;

- розробити архітектуру мовного сервера для написання природномовних текстів,
- описати та розробити протокол мовного сервера;
- побудувати програмну реалізацію системи;
- розробити опис використання програмної системи.

Потенційними користувачами даного програмного забезпечення стануть рецензенти, редактори текстів, письменники, журналісти та інші корисутвачі текстових редакторів.

1. ЗАСОБИ ДЛЯ НАПИСАННЯ ПРИРОДНОМОВНИХ ТЕКСТІВ

Природномовний текст являє собою множину слів природної мови, що використовується для спілкування між людьми.

Програмна обробка природномовних текстів виникла одразу після появи обчислювальної техніки.

Зі збільшенням продуктивності обчислювальних машин, створенням нових методів обробки інформації та їх вдосконаленням (використання нейронних мереж, створення та використання більш містких словників), зростає кількість та якість вихідної інформації, отриманої після обробки природномовних текстів.

Karen S. Jones виділяє наступні фази розвитку обробки природних текстів [1]:

- 1940-1960 — машинний переклад;
- 1970-1980 — використання штучного інтелекту для розуміння мови;
- 1980-1990 — використання штучного інтелекту для обробки мови;
- 1990-наш час — спрощення текстів для виокремлення важливих одиниць.

Важливими задачами обробки природномовного тексту є: пошук та заміна слів з урахуванням синтаксичних особливостей мови, знаходження морфологічних та синтаксичних помилок, виділення смислових домінант і тематичної структури, визначення формальних характеристик стилю і жанру.

Обробка природномовних текстів обчислювальною машиною дає можливість вирішити значний перелік синтаксичних, семантичних, мовленнєвих та дискурсивних завдань.

Прикладами синтаксичних завдань обробки природномовних текстів є:

- граматичне виведення — створення формальної мови з природної мови [2-3];
- морфологічний розбір [4];
- синтаксичний аналіз;

- стемінг — процес скорочення слова до основи [5-6];
- сегментація на слова — виокремлення слів тексту, є досить важливою задачею для китайської та японської мов;
- сегментація на речення.
- створення глосарію за текстом;
- визначення частин мови;
- лематизація — приведення словоформи до нормальної форми.

Прикладами семантичних завдань обробки природномовних текстів є:

- машинний переклад [7];
- розуміння природної мови;
- генерування природної мови;
- розпізнавання мови.

1.1 Особливості написання природномовних текстів

На відміну від предметно-орієнтованих мов програмування [8-10], природні мови не є формальними мовами, не мають скінченну послідовність символів, які описуються правилами певного виду, також їх граматика та алфавіт змінюються час від часу.

Текст, написаний природною мовою неможливо записати формальною мовою, тому наявні інструменти для написання текстів природною мовою досить складні, деякі задачі неможливо виконати повністю правильно.

Кожна природна мова має свої морфологічні та синтаксичні особливості, які слід враховувати при створенні інструментів для написання природномовних текстів даною мовою.

Отже, характеристиками природної мови є:

- алфавіт;
- набір правил для словотворення;
- набір синтаксичних та морфологічних правил [11-14].

Одиниця алфавіту — графема.

Ознаки графеми:

- тип знаку;
- фонетичні ознаки;
- розмір графеми;
- належність до алфавіту.

Набір правил для словотворення з використанням регулярних виразів наведений на рисунку 1.1.

Код класу	Найменування лексеми	Опис правила	Приклад
L01	Слово з малої літери	([а-я]+)	танк
L02	З великої літери	([А-Я][а-я]*)	Михаил
L03	Іншомовне слово	[a-zA-Z]+	tomahawk
L04	Неповна лексема	([а-я]+)-	полу-
L05	Слово через дефіс	([а-я]+)-([а-я]+)	полу-

Рисунок 1.1 — Можливий набір правил створення лексем для природної мови.

Правила синтаксису поділяються на:

- контекстні синтаксичні правила;
- правила виокремлення присудка і підмета;
- правила виокремлення другорядних членів речення.

Набір контекстних синтаксичних правил з використанням узгодження, керування та прилягання наведений на рисунку 1.2.

Який клас	З яким класом	рід	число	відмінок	Порядковий № гол. сл.	Синтаксичний тип	Приклад
2*	1*	+	+	+	(2)	С1	державні органи
23*	1*			2	(2)	У1	від форми
1*	1*			2	(1)	У2	генератор шуму
14*	8*				(2)	П1	швидко біг

Рисунок 1.2 — Можливий набір синтаксичних та морфологічних правил.

Набір правил для словоутворення формують лематизаційні словники, а синтаксичних правил — синтаксичні. Обробка природної мови на цих рівнях фактично зводиться до обробки даних відповідних таблиць.

В даній роботі, в якості природної мови було вибрано українську мову. Головною особливістю української природної мови є використання словотворчих та формотворчих афіксів (морфем, приєднаних до кореня), належить до типу флективних мов.

Усі природні мови поділяються на:

- кореневі (китайська, бірманська);
- аглютинативні (тюркські й фінно-угорські мови);
- флективні (українська, англійська, французька, болгарська);
- полісинтетичні (ацтекська мова).

Кореневі мови — мови, які не мають афіксів і граматичні значення виражають способом прилягання одних слів до інших або за допомогою службових слів.

В аглютинативних мовах граматичні форми й похідні слова утворюються додаванням однозначних та стандартних афіксів до незмінюваних основ слів.

У флективних мовах граматичні форми й похідні слова утворюються за допомогою додавань флексій (багатозначних закінчень).

Усі флективні мови поділяються на:

- синтетичні (українська, російська, польська, литовська);
- аналітичні (англійська, французька, болгарська).

У синтетичнах мовах відношення між словами виражаються завдяки формам слів, а в аналітичних — за допомогою службових слів і порядком розташування повнозначних слів.

В полісинтетичних мовах морфеми приєднуються разом і групують єдине ціле.

Алфавіт української мови складається з 33 літер, особливостями українського алфавіту у порівнянні з іншими кириличними є наявність букв “Ґ”, “Є” та “Ї”.

В українській мові виділяють лише два види морфем (найменша частина слова, яка складається з фонем): корені, афікси.

У залежності від позиції в слові афікси розрізняють на: префікси, постфікси, інтерфікси (афікс, що розташований між коренями складного слова), циркумфікси (одночасне поєднання префікса і суфікса) та інфікси.

В українській мові виділяються десять частин мови: 6 самостійних та 4 службових.

В українській мові іменники мають такі граматичні характеристики:

- мають один з трьох родів: чоловічий, жіночий, середній;
- змінюються за числами: одниною, множиною та двоїною;
- змінюються за відмінками.

В української мові іменник має узгодження с прикметником.

Дієслово має 4 часи (давноминулий, минулий, теперішній, майбутній), усі дієслова поділяються на два види: доконаний і недоконаний, також деякі дієслова є двовидові. У теперішньому та майбутньому часі дієслова відмінюються за особою та числом. У минулому часі — за родом та числом.

Основним способом утворення форм множини в українській мові є зовнішня флексія, іноді супроводжувана змінами кореневих приголосних.

Основним типом граматичних афіксів в українській мові є флексія (належать до постфіксів).

Флексія — закінчення, за допомогою якого створюються граматичні форми слів у певній системі словозміни змінюваних частин мови.

Українська мова має наступні національно-своєрідні морфологічні особливості: невідмінювання першого компонента у складних числівниках, які означають назви десятків, інфінітивна форма дієслова закінчується на “-ти” та інші.

Українська мова має наступні національно-своєрідні синтаксичні особливості: використання словосполучень, утворені підрядним зв’язком керування, використання конструкцій з предикативними формами на “-но”, “-то” з дієслівною власне-зв’язкою “бути” та інші.

Аналіз природномовних текстів складається з наступних етапів [15-17]:

- лексичний аналіз;
- морфологічний аналіз;

- синтаксичний аналіз;
- семантичний аналіз;
- прагматичний аналіз.

Лексичний аналіз — розбиття вхідного тексту на розділи, абзаци, речення та слова.

Морфологічний аналіз — перевірка правильності написання слів за словниками.

Синтаксичний аналіз — аналіз вхідної послідовності символів з метою розбору граматичної структури з перевіркою відповідності граматичним правилам.

Семантичний аналіз — встановлення смислової правильності синтаксичних конструкцій в реченнях.

Прагматичний аналіз — встановлення смислової правильності між абзацами, діалогами та текстами.

Найбільшою проблемою обробки природномовних текстів є проблема омонімії, оскільки неможливо використовуючи лише граматичний словник завжди правильно зіставити вхідну лексему слову, яке використав користувач. Розпізнавання вхідного тексту потребує величезних обсягів інформації, зазвичай це реалізовується у вигляді онтології [18].

Існують наступні обмеження для правильної обробки природномовних текстів:

- референційна неоднозначність;
- відмінкова неоднозначність;
- синтаксична неоднозначність;
- смислова неоднозначність.

Останнім часом, дані обмеження вдається уникати за допомогою систем штучного інтелекту [19], а сама обробка текстів зводиться до AI-повної задачі (еквівалентність обчислювальної складності цих задач створенню комп’ютерів, настільки ж розумних, як і люди).

1.2 Інструменти для написання природномовних текстів

Інструментом для написання природномовних текстів є текстовий редактор.

Текстовий редактор — програмний застосунок для написання та редагування текстів.

Більшість наявних текстових редакторів надають певний функціонал для прискорення написання текстів: розумне доповнення, підсвічування синтаксису, створення шаблонів, сортування рядків, мають розширені функції форматування тексту, наприклад вставка графіків, посилань, таблиць.

Деякі з інструментів для написання природномовних текстів забезпечують перевірку орфографії, що є досить складною для реалізації можливістю, тому не завжди виконується повністю правильно.

Перевірка орфографії — це процес пошуку слів, написаних неправильно та створення можливих корекцій. Перевірка орфографії та автокорекція широко застосовуються для таких завдань, як обробка слів і постобробка тексту з фотографії.

Більшість орфографічних систем потребують певних мовних ресурсів, таких як лексика, списки неправильних написань чи бази правил.

Для забезпечення мовної підтримки, інструменти для написання природномовних текстів використовують словники слів, які зазвичай зберігаються у реляційних або постреляційних баз даних. Якість такого інструменту в першу чергу залежить від об’єму словника та дати створення словника.

Текстові редактори зазвичай використовуються наступні типи словників:

- граматичний словник;
- етимологічний словник;
- словник синонімів.

Граматичний словник зазвичай є словником квазіфлексій [20] та містить мінімальну послідовність букв, яку необхідно відкинути від шуканого слова та

послідовність букв, яку необхідно додати, номер граматичної категорії, номер словозмінного класу. Граматичний словник приводить словоформу до початкової форми.

Недоліками наявних інструментів для написання природномовних текстів є:

- використання неактуальних граматичних словників, або повна їх відсутність;
- відсутність стандартного мовного протоколу;
- більшість з них не використовують мовний сервер;
- відсутність можливості розширення їх мовно-підтримуючого функціоналу.

Відсутність мовного сервера приводить до переліку інших побічних проблем, такі як:

- оновлення бази словника слів потребує оновлення усього інструменту для написання текстів;
- зменшення продуктивності мовного клієнта;
- низька швидкість аналізу текстів.

Перевагами відсутності мовного сервера є:

- відсутність вимоги до постійного з’єднання до мережі Інтернет;
- дані користувача не залишають його пристрій та не відправляються третім особам.

Прикладами інструментів для перевірки написання природномовних текстів є: Hunspell, Windows Spellchecker. Вони використовуються у браузерах з рушієм Chromium, та фактично використовуються при написанні листових повідомлень у них.

Веб-сервісами для перевірки орфографії є LanguageTool, Grammarly, Onlinecorrector, Languagetool.

Вказані сервіси надають лише обмежену мовну підтримку для перевірки орфографії, не підтримують створення певних правил користувачем і не є текстовими редакторами, в той час як наявні текстові редактори зазвичай не надають можливості для знаходження орфографічних та синтаксичних помилок.

1.3 Написання природномовних текстів у Visual Studio Code

Серед наявних розширень для написання природномовних текстів у Visual Studio Code на офіційному магазині розширень українською мовою, який доступний з меню VS Code, або за посиланням “https://marketplace.visualstudio.com/” на даний момент існують лише наступні: Ukrainian — Code Spell Checker, Ukrainian support for LteX та Ukrainian Support for LanguageTool.

Вказані розширення не використовують мовні сервери та потребують інсталяцію інших розширень.

Оскільки кожне з розглянутих розширень має перелік недоліків, доцільно було би створити нове розширення на базі наявних, яке вміщає в собі переваги кожного з розширень та має більш широкий функціонал та підвищену людино-машинну взаємодію.

Метою даної роботи є побудова мовного клієнта — розширення для середовища розробки Visual Studio Code для написання природномовних текстів та створення мовного сервера.

Для досягнення даної мети потрібно спроектувати та розробити наступні модулі мовного клієнта:

- модуль розумного доповнення;
- модуль діагностики;
- модуль підказок.

Мовний клієнт повинен мати наступні функціональні можливості:

- можливість одночасної роботи з декількома відкритими документами;
- можливість зміни режиму роботи модуля діагностики;
- збереження поточних налаштувань користувача до глобальних налаштувань Visual Studio Code;
- можливість зміни активної локалізації.

Висновки до розділу

Було розглянуто особливості природномовних текстів та інструментальне забезпечення для їх використання.

Кожна природна мова має свої морфологічні та синтаксичні особливості, які слід враховувати при створенні інструментів для написання природномовних текстів даною мовою.

Головною особливістю української природної мови є використання словотворчих та формотворчих афіксів.

Наявні інструменти для написання текстів природною мовою досить складні, деякі задачі неможливо виконати повністю правильно.

Було вирішено створити розширення для середовища розробки Visual Studio Code для написання природномовних текстів та створення мовного сервера.

2. АРХІТЕКТУРА ІНСТРУМЕНТУ НАПИСАННЯ ПРИРОДНОМОВНИХ ТЕКСТІВ

В якості інструменту написання природномовних текстів було вирішено використати Visual Studio Code, створений в 2015 році, який поширюється безкоштовно та має ряд переваг у порівнянні з іншими середовищами.

Однією з головних переваг VS Code є великий прикладний програмний інтерфейс з доступною документацією. Також слід зазначити, що VS Code є новітньою розробкою, тому оновлюється на постійній основі.

2.1 Базові засоби розширення Visual Studio Code

Visual Studio Code — високопродуктивний редактор коду, який містить служби для мов програмування та потужне середовище розробки. Розроблений на базі іншого редактора коду — Atom.

Visual Studio Code має вбудовану підтримку мов JavaScript та TypeScript, вбудований термінал, який можна використовувати для виконання команд оболонки, вбудовану систему керування версій та включає підтримку систему керування версій Git. Багато інших систем керування версій доступні через розширення на VS Code Marketplace.

Visual Studio Code використовує Node.js. Node.js — це платформа для створення швидких і масштабованих серверних додатків за допомогою JavaScript. Node.js — це середовище виконання, а NPM — менеджер пакетів для модулів Node.js. Щоб запустити додаток Node.js, на комп’ютері потрібно бути встановлене середовище виконання Node.js.

VS Code містить інші програмні продукти:

- Yeoman — інструмент створення додатків;
- Aspnet генератор — генератор Yeoman для додатків ASP.NET Core;

- Hottowel генератор — генератор Yeoman для швидкого створення AngularJS-додатків;
- Express — фреймворк для Node JS застосунків;
- Gulp - система виконання завдань;
- Mocha — фреймворк на JavaScript для тестування, що працює на Node.js;
- Bower — менеджер пакетів клієнтської сторони.

Всі розширення [21] VS Code мають загальну модель реєстрації, активації (завантаження) та доступу до інтерфейсу VS Code.

Існують два спеціальних типи розширень VS Code, мовних серверів і налагоджувачів.

Основними засобами VS Code є:

- розширення — базовий будівельний модуль;
- сервер мови — основний модуль обробки тексту певною мовою;
- дебагери — зовнішній налагоджувач, під'єднаний через адаптер.

Загальна архітектура розширення для VS Code подана на рисунку 2.1.

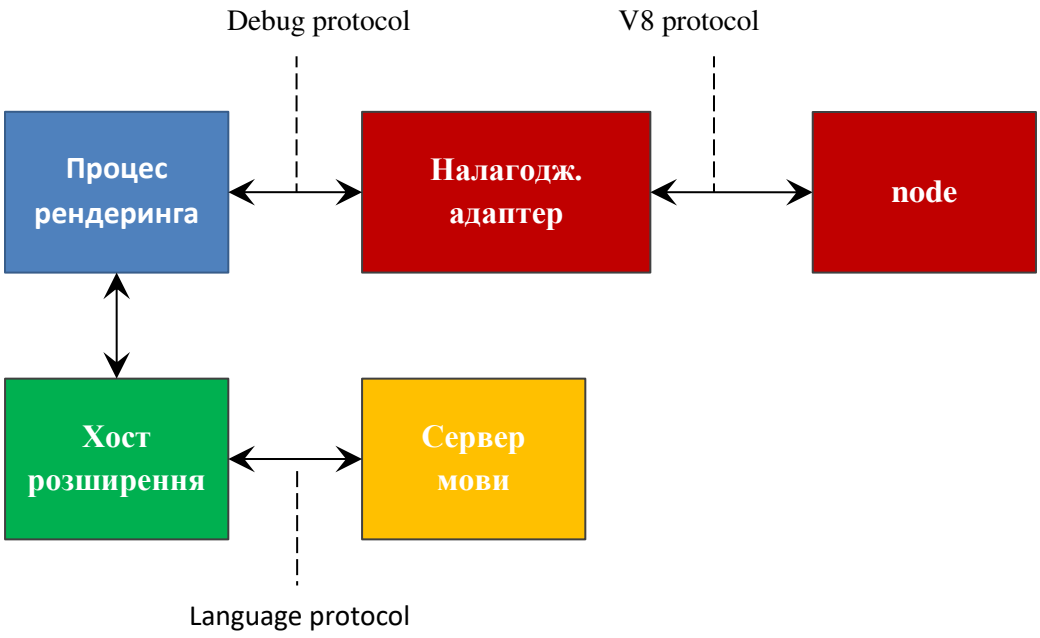

Рисунок 2.1 — Архітектура VS Code розширення.

VS Code та розширення для неї, використовують для рендерингу фреймворк Electron, який надає можливість створювати десктопні програми за допомогою веб – програмування, який використовує рушій Chromium.

Усі розширення після активації виконуються в процесі хосту спільного доступу, який не дає можливості для прямого доступу до об’єктної моделі Visual Studio Code. Дана реалізація розширень у виділеному хості має ряд недоліків та обмежує розробників розширень, проте цей окремий процес для розширень гарантує, що VS Code буде правильно реагувати на усі події та буде надавати чуткий інтерфейс користувачу.

Окремий хост та ізоляція розширення забезпечує стабільність VS Code, як програмного продукту, внаслідок чого, користувач повністю контролює роботу VS Code, розширення не може заборонити відкривання певного файлу, створення чи видалення, не може виконати включення чи виключення інших розширень, не може виконати ті дії, які не бажає користувач. Хост розширення — це процес Node.js, який відкриває API VS Code для розробників розширень.

VS Code надає підтримку налагодження для розширень, що працюють всередині хосту розширення.

VS Code активує розширення настільки пізно, наскільки це можливо, для забезпечення правильного завантаження VS Code та забезпечення режиму максимальної потужності. Розширення VS Code, які виключені в активній сесії не завантажуються взагалі, і тому вони не споживають оперативну пам'ять. Для забезпечення лінивого завантаження VS Code визначає події, які ініціюють завантаження розширення. Це може бути відкриття певного типу файлу, команда розширення, яка вибирається за допомогою меню команд або комбінації клавіш.

Розширення включають підтримку:

- активація розширення;
- редактор — робота з контентом редактора, наприклад читання та маніпулювання текстом;
- робоча область — відкриті редактори, панель статусу, інформаційні повідомлення та інше;

- події — підключення до подій життєвого циклу редактора, таких як: відкриття, закриття, зміна тощо;
- розвинуте редагування — створення провайдерів для підтримки багатьох мов, включаючи IntelliSense, Peek, Hover.

Можливості розширень в VS Code:

- підсвічування синтаксису;
- розумні доповнення (IntelliSense);
- діагностика коду та виправлення;
- показ інформації про сигнатури функцій;
- навігація коду (перейти до визначення, знайти всі посилання);
- налагодження;
- форматування коду;
- рефакторинг.

Загальним паттерном створення розширення у VS Code є виконання коду розширення в окремому процесі, який зв’язується з VS Code через протокол. Прикладами цього в VS Code є мовні сервери та налагоджувальні адаптери. Як правило, цей протокол використовує stdin / stdout для взаємодії між процесами, використовуючи JSON.

Використання окремих процесів забезпечує хороші межі розділення виконання коду, що допомагає VS Code зберегти стабільність основного редактора.

2.1.1 Сервер мови

Сервери мови дозволяють створювати виділений процес для розширення. Перевірка та обробка певної мови може займати великі ресурси, особливо коли перевірка виконується одночасно для декількох файлів і створюються дерева лексичного та синтаксичного розбору. Для того, щоб мінімізувати втрату продуктивності, мовні сервери в VS Code виконуються в окремому процесі.

Мовні сервери можуть бути написані будь-якими мовами, проте VS Code надає прикладний інтерфейс лише для мов Javascript та Typescript.

Зазвичай сервер мови використовується, коли розширення працює над інтенсивними завданнями CPU або IO, які можуть сповільнити роботу інших розширень.

VS Code дозволяє застосувати сервер мови як автономний сервер, що реалізує протокол мовного сервера, або безпосередньо реєструвати постачальників у методі активування розширення:

- protocol language server protocol;
- direct implementation.

Для створення мовного сервера на мові Typescript можна використати офіційну бібліотеку “vscode-languageserver”, яка надає інтерфейси IConnection IPCMessageReader, IPCMessageWriter. IConnection використовує Node JS IPC для міжпроцесорної взаємодії (рисунок 2.2). За допомогою методу listen класу TexDocument, менеджер документів реєструє підписку на зміну, відкриття нового та закриття документів для переданого до методу параметру — підключення.

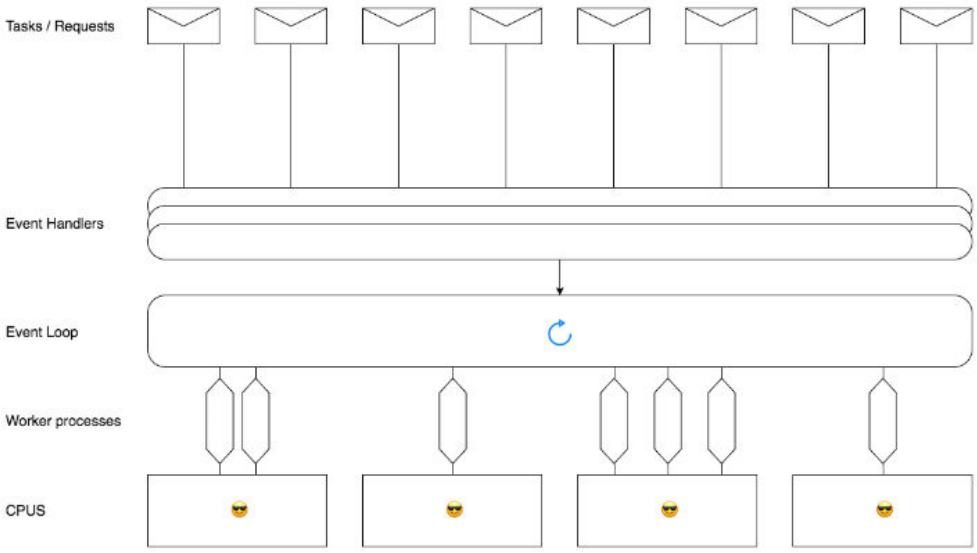

Рисунок 2.2 – Node JS IPC.

У даній роботі, в якості мовного сервера використовується явна реєстрація провайдерів у методі активації розширення — direct implementation.

Функція активації розширення — activate, яка є експортованою, спочатку завантажує та обробляє збережені налаштування користувача, реєструє команди дій

користувача, реєструє провайдери, реєструє події, налаштовує інтерфейс користувача та робить первинну обробку відкритого файлу (парсинг).

2.1.3 Маніфест розширення

Кожне розширення VS Code потребує файл маніфесту (опис розширення). Файл маніфесту — це файл package.json, який знаходиться у корені каталогу розширення. Він забезпечує VS Code всією необхідною інформацію, яка в першу чергу також використовується при опублікуванні розширення в магазині розширень.

Маніфест даного розширення включає інформацію про назву розширення, опис, версію, категорію, події які ініціюють включення розширення, список тем та команди, список налаштувань користувача, список клавіш швидкого доступу, інформацію про ліцензію та залежності від інших бібліотек, посилання на репозиторій, мінімальну підтримуючу версію VS Code, категорії до яких відноситься розроблене розширення та події, які активують розширення, шлях до виконуваного файлу.

Розширення, які відносяться до однієї категорії групуються разом у магазині розширень, що прискорює фільтрацію та пошук. Інформація про категорії може включати наступні значення: Programming Languages, Snippets, Linters, Themes, Debuggers, Formatters, Keymaps, SCM Providers, Other, Extension Packs, Language Packs. Дане розширення відноситься до наступних категорій: Languages, Themes, Other.

Список налаштувань користувача (ключ contributes.configuration) надає інформацію про налаштування за замовчуванням робочої області та глобальних налаштувань користувача. Дане розширення вказує глобальні налаштування локалізації — “en-US” та режиму роботи модулю діагностики — “true” (активований).

Ключ contributes.keybindings вказує перелік клавіш швидкого доступу з відповідної приєднаною командою користувача та можливою умовою виконання. Список клавіш швидкого доступу даного розширення вказаний у таблиці 2.1.

Таблиця 2.1.

Назва команди	Опис команди	Клавіша швидкого доступу
extension.aboutVersion	Version	ctrl+f0
extension.startWork	Запуск розширення	ctrl+f1

Слід зазначити, існує можливість, що у Visual Studio Code, може бути встановлене інше розширення, яке має команди з тими ж самими вказаними клавішами швидкого доступу. Тоді виникає деякий конфлікт, оскільки клавіша прив'язана одночасно до двох чи більше команд, що може спричинити неправильну поведінку кожного з розширень.

Меню налаштувань швидкого доступу надає можливість користувачу дізнатися про перелік наявних конфліктів та змінити комбінацію на іншу.

Файл конфігурації мови включає інформацію про типи коментарів, автозакриваючі символи, лексеми, які автоматично виконують форматування тексту та пари лексем, які автоматично підсвічуються.

Оскільки хост розширення — це процес Node.js, розробник може використовувати API Node.js у своїх розширеннях, або використовувати наявні модулі Node.js при розробці розширення. В цьому випадку розробник визначає свої модульні залежності в package.json (ключі dependencies та devDependencies), і використовує npm для встановлення Node.js модуля. Приклад використання залежностей наведений на рисунку 2.3.

```
"dependencies": {
    "mssql": "^4.0.4",
    "mysql2": "^1.5.1"
},
"devDependencies": {
    "typescript": "^2.6.1",
    "vscode": "^1.1.6",
    "@types/node": "^7.0.43",
    "@types/mocha": "^2.2.42"
},
```

Рисунок 2.3 – Використання маніфесту розширення.

2.1.4 Особливості API Visual Studio Code.

API VS Code представляє асинхронні операції з “обіцянками”. З розширень можна повернути будь-який тип “обіцянок”, як ES6, WinJS, A + тощо.

Незалежність від конкретної бібліотеки з “обіцянкою” виражається в API за допомогою типу-інтерфейсу Thenable. Thenable інтерфейс повертає об’єкт з методом then. Thenable інтерфейс повертається у кожному провайдері розширення.

У більшості випадків використання обіцянок є необов'язковим, і коли VS Code викликає розширення, він може явно повернути результат. Коли використання обіцянки є необов'язковим, API це вказує, повертаючи обидва типи:

provideNumber(): number | Thenable<number>.

Часто операції VS Code мають можливість змінюватися, перш ніж операції можуть закінчитися. Наприклад, починається обчислення модулю розумного доповнення, і користувач продовжує вводити текст, в результаті чого результат цієї операції стає застарілим.

API, які реалізовують таку поведінку, отримують токен CancellationToken, за яким розробник може перевірити скасування (isCancellationRequested) або отримувати сповіщення, коли відбувається скасування (onCancellationRequested). Токен скасування зазвичай є останнім параметром виклику функції та є необов’язковим.

API VS Code використовує шаблон звільнення ресурсів (Disposable pattern), отриманих з VS Code. Це стосується прослуховування подій, команд, взаємодії з інтерфейсом користувача та різними частинами мови.

Наприклад, функція setStatusBarMessage (значення: string) повертає Disposable об’єкт, який після виклику методу dispose видаляє повідомлення з панелі показу статусу.

Події в API VS Code є функціями, які підписуються на певну подію DOM. Функції, які повертаються після підписання, мають тип Disposable і можуть автоматично відписатися після виконання методу dispose. Приклад:

```
var subscription = fsWatcher.onDidDelete (listener);
subscription.dispose ();
```

Назви подій мають on[Will|Did]VerbNoun? паттерн. Назва сигналізує, що подія відбудеться (onWill) або вже відбулася (onDid), що сталося (дієслово) і контекст (іменник), якщо це не є очевидним з контексту.

Прикладом VS Code API є window.onDidChangeActiveTextEditor, що відбувається після зміни активного текстового документа.

В даній роботі виконується підписка на:

- відкривання документу – onDidOpenTextDocument;
- зміна контенту документа – onDidChangeTextDocument;
- зміна активного документа – onDidChangeActiveTextEditor.

При виконанні будь-якої події із списку вказаного вище, у даному роширенні ініціюється виконання лексичного аналізатора.

2.1.5 Публікація розширення

Для публікації розширення VS Code (включаючи розроблене), використовується модуль vsce (The Visual Studio Code Extension Manager). VSCE — це інструмент командного рядка, який використовується для публікації розширень на Extension Marketplace. Користувач також можете завантажувати розширення локально (як vsix файл).

Через проблеми безпеки, vsce не публікує розширення, що містять SVG зображення.

Visual Studio Code використовує служби Visual Studio Team Services для своїх служб Marketplace. Це означає, що через цю службу здійснюється аутентифікація, хостинг та управління розширеннями. VSCE публікує розширення за допомогою особистих токенів доступу. Щоб опублікувати розширення, потрібно створити принаймні один токен доступу.

Для публікації розширення використовується команда vsce publish, для видалення пакету — vsce unpublish.

Для створення локального VSIX файлу використовується команда vsce package, яка поміщає створений файл у корінь проекту розширення.

Перед публікацією розширення потрібно вказати мінімальну підтримуючу версію VS Code у файлі маніфесту — ключ engines.vscode. Граматика значення співпадає з семантикою ведення версій пакету Node.js та підтримує квантифікатори ^, ~, *.

В розробленому розширенні для публікації також вказані категорії, до яких відноситься дане розширення (рисунок 2.4).

```
"publisher": "Goldych",
"engines": {
    "vscode": "^1.19.0"
},
"categories": [
    "Languages",
    "Other"
],
```

Рисунок 2.4 — Маніфест розширення для публікації.

2.2 Архітектура середовища написання природномовних текстів

Для підтримки написання природномовних текстів з дотриманням функціональних вимог було вирішено розробити:

- мовний клієнт;
- мовний сервер;
- протокол мовного сервера.

Мовний клієнт повинен використовувати протокол мовного сервера для зв’язку з мовним сервером.

Мовний клієнт повинен мати наступні модулі:

- модуль доповнення тексту;
- модуль діагностики;

- модуль підказок.

Модуль діагностики повинен надавати користувачу інформацію про помилки вводу тексту запиту, наприклад невірний символ. Вхідними даними є природномовний текст. Вихідними даними є текст помилки, індекси рядків та колонок початку та кінця помилки, можливі автоматичні виправлення.

Модуль розумного доповнення повинен надавати інформацію про список можливих доповнень, які починаються з тексту, який був введеним користувачем, що забезпечить прискорення написання тексту. Вхідними даними є природномовний текст. Вихідними даними є список лексем.

Модуль підказок повинен надавати опис відповідної лексеми. Вхідними даними є природномовний текст, індекс наведеного на лексему курсору. Вихідними даними є опис лексеми.

Висновки до розділу

В якості інструменту написання природномовних текстів було вирішено використати Visual Studio Code, розширення для якого складається з модулей, які є інструментальними засобами, а саме: модуль розумного доповнення, модуль підсвічування тексту, модуль проведення діагностики, модуль підказок.

Модулі взаємодіють між собою явно за допомогою інтерфейсів або неявно за допомогою виконання заімплементованих методів, безпосередньо ініційованими VS Code, фактично — це дії користувача з розробленим розширенням.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

В якості мови програмування для реалізації мовного клієнта було вирішено використати мову Typescript.

Мова програмування Typescript представлена Microsoft восени 2012, поширюється під ліцензією Apache, компілятор транслює код TypeScript в представлення JavaScript.

Typescript розширює можливості Javascript та має наступні переваги: сувора типізація даних, являю собою повністю об’єктно-орієнтовану мову програмування, велика читабельність коду, зворотна сумісність з JavaScript, широка підтримка середовищами розробки, написаний код поширюється у вигляді модулів, підтримка узагальнених типів.

В якості середовища розробки мовного клієнта використовується Visual Studio Code.

Для створення каркасу мовного клієнта був використаний генератор Yeoman [22]. Генератор, за допомогою інтерфейсу командного рядка, дозволяє створити каркас для нового розширення або створити готові до використання розширення для мов, тем або фрагментів на основі наявних файлів TextMate.

Можливості генератора Yeoman:

- створення нового розширення (TypeScript);
- створення нового розширення (JavaScript);
- створення нового розширення з темою для IDE;
- створення нового розширення з шаблонами;
- створення підтримки нової мови з модулем підсвічування.

Генератором Yeoman була створена наступна структура VS Code розширення:

- .gitignore файл з списком виключень для системи керування версіями;
- .vscodeignore з списком виключень при розгортанні розширення;
- README.md файл документації;

- src папку з файлом extension.ts, який вміщає точку входу розширення;
- out папку, яка вміщає скомпільований код.
- node_modules папку з залежними модулями;
- package.json файл, який є маніфестом розширення.

В якості мови програмування для реалізації мовного сервера було вирішено використати мову C# 7 та платформу .NET 4.7.

Microsoft .NET – платформа від Microsoft, випущена у 2002 році, яка використовує загальномовне виконуюче середовище CLR.

CLR транслює початковий код в байт-код на мові IL, під час запуску програми відбувається JIT (Just-in-time) компіляція, де інструкції IL конвертуються в машинні інструкції. JIT компіляція відбувається кожного разу та займає певний час, тому була надана можливість створення машинного коду завчасно, за допомогою утиліти ngen.exe, проте слід враховувати що це має ряд недоліків. Усі бібліотеки .NET для прискорення роботи проходять попередню компіляцію до та зберігаються у глобальному кеші збірок (GAC).

Використання проміжної мови дозволяє розгортати програмний продукт маючи лише одну конфігурацію AnyCPU для різних архітектур процесора (x86, x64), оскільки в даному випадку JIT компіляція буде виконуватись з врахуванням поточної архітектури процесора.

Компілятор .NET-мови створює метадані і зберігає їх в збірку, яка містить CIL. Збіркою є виконуючий файл “.exe ”або бібліотека “.dll”. Метадані описують всі класи та члени класів, а також класи та члени класів, які поточна збірка викликає з іншої збірки, використовуються при використанні рефлексії.

Microsoft .NET підтримує автоматичний збір сміття за допомогою використання лічильників, управління потоками (як на рівні фізичного потоку так і на рівні асинхронної задачі), обробку виключних ситуацій.

Microsoft .NET надає велику кількість бібліотек, документацію та оновлюється на постійній основі.

Мова C++ CLI, яка підтримує CLI, дозволяє зв'язувати збірки написані на .NET з мовою C++, за рахунок створення обгорток на C++ CLI.

Мовний сервер використовує набір бібліотек для роботи з словником, які працюють платформі .NET, отже відсутня необхідність створювати проксі бібліотеки для роботи зі словником між різними платформами виконання.

В якості середовища розробки мовного сервера використовується Visual Studio 17.

Visual Studio, підтримує можливість використання доповнень, надає систему керування пакунками NuGet від Microsoft, яка в першу чергу використовується для .NET.

3.1 Мовний клієнт

Мовний клієнт — текстовий редактор, призначений для створення й зміни текстових файлів через графічний інтерфейс користувача.

Мовний клієнт отримує повідомлення про зміну тексту користувача, активного документа, місцеположення каретки вводу та передає відповідну інформацію на мовний сервер по спеціально розробленому мовному протоколу.

Мовний клієнт отримує повідомлення від мовного сервера та надає оброблені дані користувачу через графічний інтерфейс користувача. Фактично з використанням мовного сервера на стороні клієнта залишається лише одна вимога— наявність стабільного швидкого доступу до мережі Інтернет.

Мовними клієнтами можуть бути нативні додатки, веб-застосунки, гібридні додатки (зазвичай це використання веб-застосунків всередині нативних додатків).

Зазвичай розробники мовних клієнтів надають можливість для створення розширень для їх продукту, та викладення їх на маркет розширень. Реалізація мовних розширень залежить від операційної системи та архітектури мовного клієнта, наприклад, розширення для Microsoft Outlook для Windows використовують COM технологію.

В даній роботі в якості мовного клієнта вибраний VS Code, створюючи розширення для нього, архітектура якого наведена на рисунку 3.1.

Реалізований мовний клієнт складається з провайдерів підтримки мови, процесорів документів, модулей для ASN кодування і декодування, обгортки над TCP та процесора TCP повідомлень.

Задля мінімізації часових витрат написання текстів та забезпечення мовної підтримки було розроблено наступні мовні провайдери:

- модуль доповнення тексту;
- модуль діагностики;
- модуль підказок.

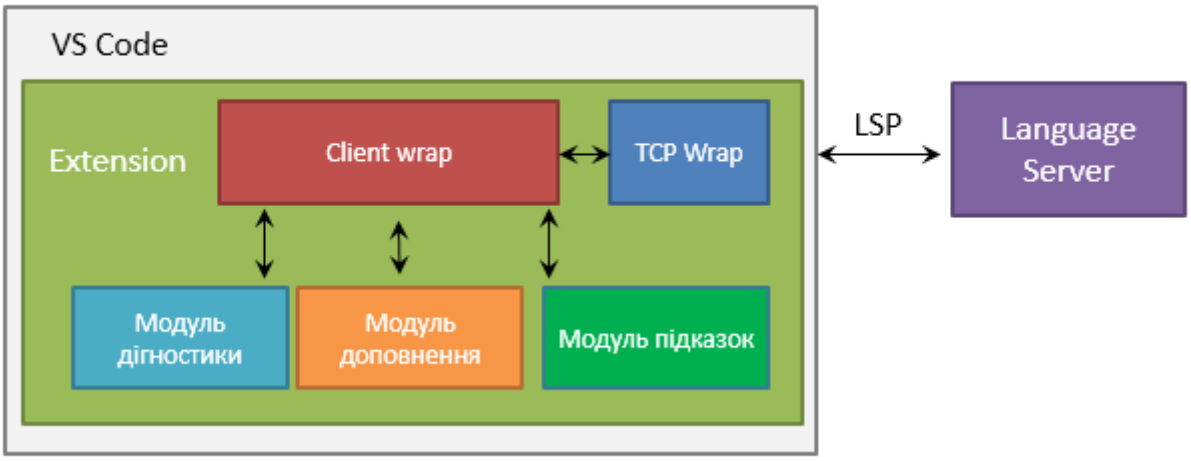

Рисунок 3.1 — Архітектура мовного клієнта.

Модулі діагностики, доповнення та підказок взаємодіють з відповідним процесором документу через посилання на vscode.TextDocument. Процесори документів зберігаються у обгортці для роботи з клієнтом (Client wrap), це необхідно для забезпечення одночасної роботи з декількома відкритими документами у Visual Studio Code.

Процесор TCP повідомлень викликає відповідні методи обгортки над клієнтом в залежності від типу повідомлення.

Оскільки програмний продукт повинен мати можливість зміни активної локалізації та збереження поточних налаштувань було вирішено розробити менеджер налаштувань та менеджер локалізації.

3.1.1 Модуль діагностики

Модуль діагностики надає список помилок (проблем) у тексті. Модуль діагностики отримує необхідні дані для відображення від мовного сервера.

Приклад роботи модуля діагностики показаний на рисунку 3.2.

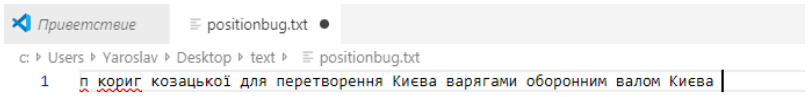

Рисунок 3.2 — Робота модуля діагностики.

На відміну від інших модулей, які є інструментальними засобами, модуль діагностики не імплементує провайдер, він підписує до контексту VS Code колекцію vscode.DiagnosticCollection. Після чого в колекцію неявно додається об’єкт Diagnostic, який вміщає інформацію про тип діагностики, повідомлення діагностики та діапазон — номери рядків та колонок тексту.

Типи підтримуючих типів діагностики на стороні клієнта:

- DiagnosticSeverity.Error;
- DiagnosticSeverity.Warning;
- DiagnosticSeverity.Information;
- DiagnosticSeverity.Hint.

Загальна кількість помилок модуля діагностики надається в графічному інтерфейсі VS Code у статус панелі.

Для видалення помилок тексту використовується видалення відповідного елементу Diagnostic з колекції помилок.

Реалізований модуль надає можливість виправляти помилки автоматично за допомогою “Code Actions” функціоналу, а саме змінювати неправильно написані слова на найбільш імовірні, які надаються мовним сервером.

Якщо дії доступні, поруч із помилкою чи попередженням з’являється лампочка. Коли користувач натискає на лампочку, з’являється список доступних дій з текстом. Для цього імплементується CodeActionProvider інтерфейс, а саме provideCodeActions метод.

3.1.2 Модуль розумного доповнення

Модуль розумного доповнення надає інформацію про контекстно-залежні пропозиції для користувача. Модуль діагностики отримує необхідні дані для відображення від мовного сервера, надсилаючи йому місцеположення каретки вводу.

Для реєстрації модуля у розширенні, що має виділений сервер мови потрібно у методі ініціалізації з’єднання повернути об’єкт, який має у ключі capabilities ключ completionProvider з значенням resolveProvider:true.

Для реєстрації модуля у розширення з явною імплементацією провайдерів потрібно імплементувати інтерфейс vscode.CompletionItemProvider, а саме метод provideCompletionItems, що і використовується у даній реалізації мовного клієнта.

При додаванні модуля до контексту розширення додатково були вказані тригер-символи “*” для запуску модуля.

Кожного разу коли користувач вводить букви або тригер-символи, викликається метод provideCompletionItems, який повертає “обіцянку”, значенням якого є масив vscode.CompletionItem. CompletionItem складається з назви, типу, тексту по якому відбувається фільтрація автоматично середовищем VS Code, тексту який вставляється до редактора документу, документації.

В даній роботі у методі provideCompletionItems повертається масив CompletionItem, де поля “insertText”, “filterText”, “documentation” та “label” встановлюються значеннями, отриманих від мовного сервера. Для поля “documentation” використовується об’єкти типу MarkdownString, який є реалізацією синтаксису MarkDownSyntax (полегшеного синтаксису розмітки даних) та мови розмітки Setext.

Модуль розумного доповнення дозволяє значно прискорити час вводу лексеми. Чим більша довжина лексеми, тим більше час, який модуль розумного доповнення дозволяє зменшити.

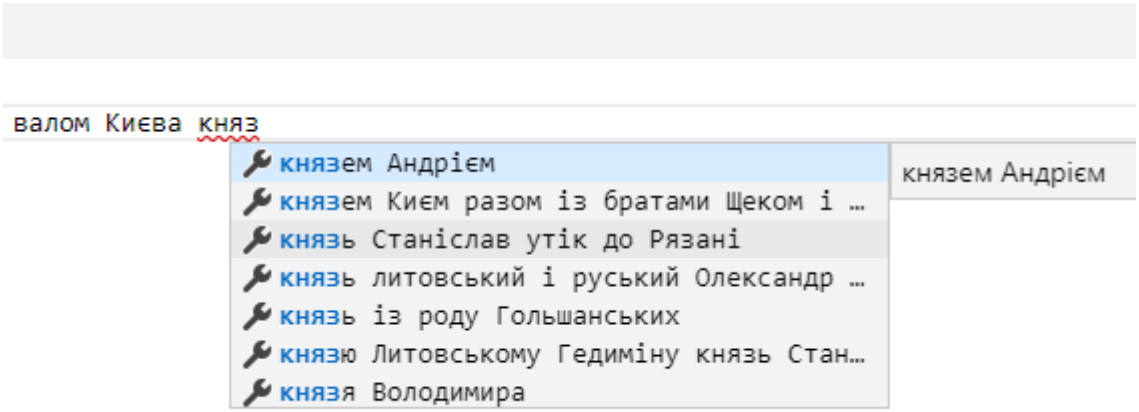

Рисунок 3.3 — Робота модуля розумного доповнення.

3.1.3 Модуль підказки

Модуль підказки надає додаткову інформацію про лексему або помилку модуля діагностики при наведенні курсору маніпулятора типу “миша” на відповідну лексему або помилку.

Повідомлення, яке надається модулем діагностики, реалізується автоматично засобами VS Code, для цього лише необхідно вказати правильний діапазон початкового та кінцевого індексів помилки тексту та відповідне повідомлення.

Повідомлення іншого типу, розраховується вручну, за допомогою масиву лексем (токенів) запиту. З масиву лексем, береться та лексема, індекси якої вміщають в себе індекс наведеного курсора. Для цього було створений клас HoverProvider, який імплементує інтерфейс HoverProvider, а саме реалізацію методу provideHover().

3.1.4 Людино-машинна взаємодія

Задля забезпечення максимальної продуктивності та максимальної швидкості програмного застосунку слід підтримувати на достатньо-високому рівні коефіцієнт людино-машинної взаємодії.

Людино-машинна взаємодія [23-24] являє собою якісну характеристику взаємодії між користувачем та комп’ютером. Це може бути взаємодія як і на рівні користувацького інтерфейсу (GUI), так і на апаратному рівні.

Критерії людино-машинної взаємодії діляться на:

- швидкість роботи з інтерфейсом;
- напруженість праці;
- кількість помилок під час роботи;
- швидкість навчання;
- продуктивність роботи.

Швидкість роботи з інтерфейсом залежить від:

- витраченого часу на аналіз вихідної інформації;
- витраченого часу на розумову діяльність;
- витраченого часу на механічну дію;
- витраченого часу зворотний зв’язок від системи.

Тривалість розумової діяльності залежить від:

- формування мети діяльності над системою;
- визначення послідовності команд;
- виконання певної сукупності команд;
- оцінки отриманого результату;
- аналізу стану системи;
- інтерпретації поточного стану системи.

Прямим чином програмна система не може покращити швидкість формування мети діяльності користувача, проте вона може значно покращити визначення послідовності команд та швидкість їх виконання.

Задля цього, усі дії, які може робити користувач над системою, були зібрані в меню команд VS Code. Перевагою цього є те, що користувач має змогу бачити повний перелік дій, команд в одному меню. Недоліком є те, що крім команд даного розширення, користувачу доступні усі команди усіх встановлених розширень для VS Code, навіть ті, які є у виключеному режимі, це є побічний ефект того, що усі

розширення виконуються в процесі спільного хоста. Тому, якщо бажана дія користувачем не виконувалась недавно або не виконувалась взагалі, ці команди будуть відображатись останніми у списку команд.

Тривалість механічних дій користувача в даному розширенні повністю залежить від VS Code. Наприклад, VS Code дозволяє при відкритті вікна вводу команд обрати команду за допомогою клавіатури (використання стрілок для вибору команд та “Enter” для підтвердження дії), що прискорює в 3-5 раз час, у порівнянні з маніпулятором типу “миша”.

Також, починаючи з версії 1.21.1 VS Code, користувач має змогу налаштувати місце відкриття вікна вводу команд.

Слід зазначити, що у різних версіях VS Code комбінації клавіш по замовчуванню можуть відрізнятися один від одного, тому рекомендовано переназначити комбінації на інші.

Для того, щоб показати, що середовище розробки реагує на вхідні дані та виконує певну операцію, VS Code показує деякий індикатор виконання цього процесу. Наприклад, на етапі завантаження даних з модуля підказок, VS Code може показати вікно з написом “Завантаження” та надає інформацію користувачу про те, що реагує на його дії.

Найбільшим недоліком даного розширення та розширень для VS Code взагалі є те, що оскільки розширення працюють у окремому процесі спільного хосту та завантажуються якомога пізніше, до тих пір, поки завантаження успішно активується та виконає функцію активації, користувач, при спробі виконати певну команду, наприклад отримати підказку чи використати інші можливості, які забезпечує розширення, отримає помилку (рисунок 3.4) або взагалі нічого не отримає.

Тому дане розширення не забезпечує необхідну тривалість зворотнього зв’язку системи на етапі завантаження середовища розробки, і користувачу необхідно дочекатися повного завантаження розширення.

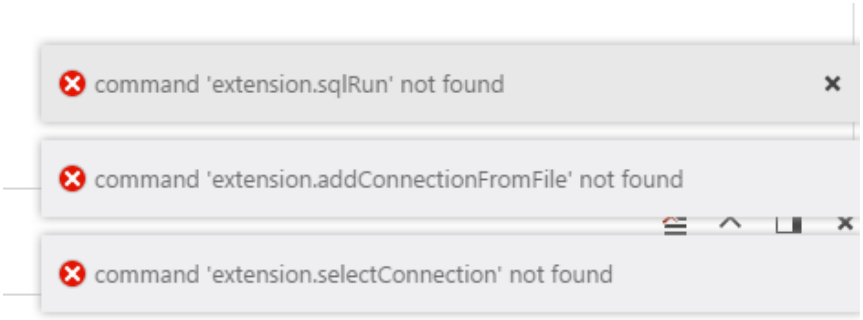

Рисунок 3.4 — Помилка при виборі команди.

Повне завантаження розширення може займати 5-20 секунд в залежності від продуктивності процесора, пріоритету процесу та інших причин.

Великий показник людино-машинної взаємодії визначає невелику кількість людських помилок.

Людська помилка — діяльність користувача, що не співпадає з його безпосередньою метою діяльності.

Людські помилки при написанні та використанні програмного забезпечення діляться на наступні типи:

- помилки, через хибне або недостатнє знання предметної області;
- описки;
- помилки, викликані хибним аналізом показань системи;
- моторні помилки.

Помилки, викликані хибним знанням про предметну область, легко вирішуються за допомогою навчання користувача в процесі користування з системою або в процесі вивчання довідкового матеріалу. Розширення забезпечує швидке навчання за допомогою однотипності та групування дій.

Описки залежать від використовуваного маніпулятора та від тривалості інтелектуальної роботи користувача. В даному розширенні при виникненні помилок автоматично спрацьовує модуль діагностики, який забезпечує правильність вводу запиту, і користувач завжди має доступ до списку та кількості помилок.

Помилки, викликані хибним аналізом системи характеризують стан, при якому бачення активного стану систему користувача не співпадає з реальним станом системи.

Прикладом є вибір вже активної мови в меню налаштувань, тому для активної мови надається відповідне додаткове повідомлення в меню.

Також слід зазначити, що для підвищення показника людино–машинної взамодії в даному розширенні використовується збереження поточних налаштувань до глобальних налаштувань, що забезпечує автоматизацію дій та зменшення фізичних дій користувача.

3.2 Мовний сервер

Мовний сервер (рисунок 3.5) забезпечує функціональну підтримку підсистеми обробки текстів. Завдяки цьому, зменшується навантаження на мовні клієнти, створюється окремий модуль, що надає однаковий функціонал різним мовним клієнтам.

Мовний сервер обмінюється повідомленнями з мовними клієнтами за допомогою протоколу мовного сервера.

Мовний сервер підтримує одночасну роботу декілька мовних клієнтів та підтримує одночасну обробку декілька відкритих документів на клієнтську сесію. Для цього створена підтримка на рівні протоколу мовного сервера та на всіх модулях мовного сервера. Усі повідомлення між модулями сервера та повідомлення у протоколі мовного сервера мають посилання на порядковий номер документа та сесії користувача.

Мовний сервер складається з трьох збірок:

- DocumentAnalyzer.dll – бібліотека для аналізу текстів та забезпечення підтримки природномовних текстів;
- Server.exe – консольний додаток, який обмінюється повідомленнями з клієнтом та використовує DocumentAnalyzer;

- Base.dll – бібліотека, яка надає базові типи для Server.exe та DocumentAnalyzer.

Було вирішено розробити наступні базові модулі для сервера у DocumentAnalyzer.dll:

- лексичний аналізатор;
- синтаксичний аналізатор;
- адаптер для роботи з словником;
- модуль забезпечення мовної підтримки.

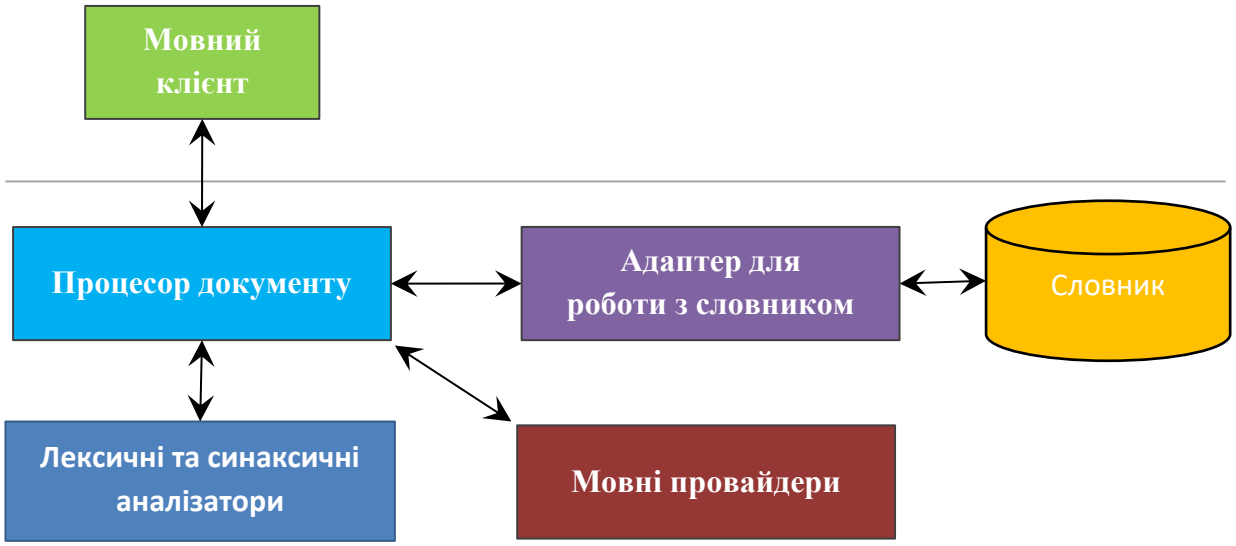

Рисунок 3.5 — Архітектура мовного сервера.

Server.exe (рисунок 3.6) складається з ID генератору для операцій, TCP Listener, ClientConnector (обробка TCP повідомлень), ASN кодерів (ASN Encoder, ASN Decoder), eClient (обгортки для роботи над клієнтом), TCP Message Wrap, та допоміжних модулей.

TCP Listener підтримує можливість зв'язку з декількома клієнтами. TCP Message Wrap отримує дані від ASN Decoder та являє собою абстракцію для розбиття повідомлень на тип повідомлення (перша частина повідомлення) та корисне навантаження повідомлення (друга частина повідомлення).

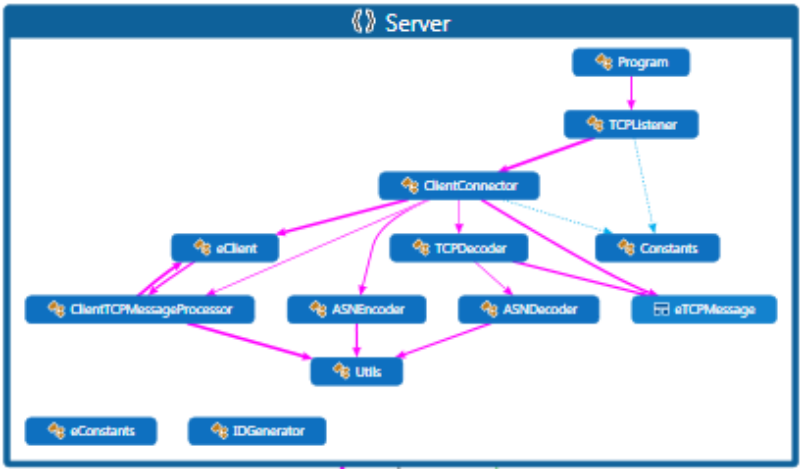

Рисунок 3.6 — Діаграма зв’язків Server.exe.

DocumentAnalyzer.exe (рисунок 3.7) складається з TokenParser (лексичного аналізатора), DiagnosticProvider (провайдер надання діагностики для тексту), DocProcessor (обгортка над документом), SpellCorrector (провайдер надання можливих дій для діагностики), DocClient (обгортка над клієнтом), класів операцій для надання мовної підтримки та допоміжних модулей.

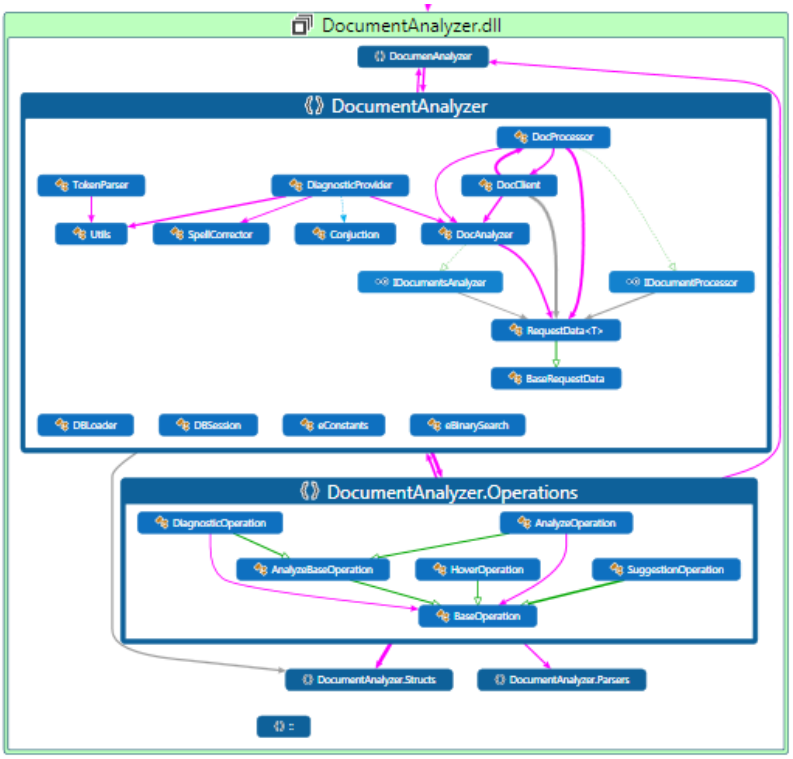

Рисунок 3.7 — Діаграма зв’язків DocumentAnalyzer.exe.

Реалізований мовний сервер має наступні недоліки:

- відсутність балансеру навантаження;
- відсутність можливості перезапуску після падіння;
- відсутність збору статистики та запису лог-інформації;
- відсутність захищеного каналу передачі даних.

3.2.1 Протокол мовного сервера

Протокол мовного сервера – LSP (Language Server Protocol), являє собою контракт, набір правил, який підтримується мовним сервером і використовується клієнтами для користування послугами сервера.

Єдиний контракт забезпечує стандартизацію повідомлень між множиною мовних клієнтів та множиною мовних серверів (рисунок 3.8).

Рисунок 3.8 — Перевага використання LSP.

Існує велика кількість різних видів протоколів та способів кодування даних.

Наприклад, широко використовується у готових реалізаціях мовних серверів протокол виклику віддалених процедур JSON-RPC [25-26] (рисунок 3.9), проте він має ряд недоліків, а саме: великий обсяг пакету через використання JSON, необхідність підтримування єдиного називання методів між клієнтами і серверами та інші. Перевагою є невелика кількість типів даних та властивостей, повідомлення має зручний вигляд. Приклад JSON RPC повідомлення: --> {"jsonrpc": "2.0",

"method": "update", "params": [1,2,3,4,5]}. JSON-RPC в якості транспортного протоколу використовує HTTP протокол.

```
--> {"method": "echo", "params": ["Hello JSON-RPC"], "id":1}
<-- {"result": "Hello JSON-RPC", "error": null, "id":1}
```

Рисунок 3.9 — Приклад JSON RPC протоколу.

Для зменшення обсягів TCP пакету, збільшення корисної пропускної здатності каналу було вирішено створити новий протокол. В якості транспортного проколу обрано TCP протокол, тому що він має наступні переваги на відміну від UDP (User Datagram Protocol):

- гарантує доставку даних до призначення;
- забезпечує широкі механізми перевірки помилок;
- гарантує послідовність даних;
- можливість повторної передачі втрачених пакетів;
- орієнтованість на використання з’єднання.

Пакет повторно передається лише протягом двох подій: коли відправник отримує три дублікати підтвердження (ACK) або після закінчення таймера retransmission timer, припускаючи, що він втрачений по шляху проходження.

В якості транспортного протоколу можна було би використати UDP протокол оскільки він швидше, простіше та ефективніше, що досить важливо для надання швидкої мовної підтримки текстів, ніж TCP, але для цього потрібно було розробляти додатковий функціонал для перевірки помилок.

В якості виду кодування даних вибрано DER [27-29] кодування. DER імплементує ASN.1 Abstract Syntax Notation One синтаксис та задовільняє X.690 стандарт ASN.1 представлення даних при їх передачі.

Кожен елемент даних кодується наступною послідовністю: ідентифікатор типу, опис довжини, фактичні елементи даних та, де необхідно, маркер кінця вмісту (рисунок 3.10).

Identifier octets *Type*	Length octets *Length*	Contents octets *Value*

Рисунок 3.10 — Кодування даних протоколу.

Структура ідентифікатора показана на рисунку 3.11.

Octet 1								**Octet 2 onwards**							
8	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Tag class		P/C	Tag number (0–30)					N/A							
			31					More	Tag number						

Рисунок 3.11 — Кодування типу.

Біти 8 і 7 визначають клас даних, в даній роботі вони дорівнюють 0 (типи, які визначені тільки в X.690 і мають однаковий сенс у всіх додатках).

Біт 6 показує чи є тип простим (дорівнює 0) або складеним (дорівнює 1). Розроблений протокол використовує лише прості типи даних. Складені типи серіалізуються та десеріалізуються у послідовності на рівні процесорів TCP повідомлень.

Біти 5 - 1 означають тег даних. Розроблений протокол використовує наступні типи даних ASN.1:

- INTEGER = 0x2 (число);
- UTF8String = 0xC (рядок);
- SEQUENCE = 0x10 (послідовність);
- OCTET_STRING = 0x4 (масив байтів).

В реалізованому протоколі складені типи кодуються наступним чином:

- vscode.Range – SEQUENCE {Start, End};
- vscode.Position – SEQUENCE {Line, Character};
- Diagnostic – SEQUENCE {Range, Message, Severity, PossibleCorrection};

- BaseRequestData – SEQUENCE {ReqId, Data}.

У випадку, якщо довжина блоку даних для збереження відомі та довжина не перевищує 127 октетів (байт), то вона просто записується у відповідний октет довжини, біт 8 першого октету встановлюється в 0. Така форма представлення октетів називається короткою формою (DEFINITE_SHORT).

У випадку, якщо довжина блоку даних для збереження відомі та довжина більше 127 байт, то: в біти другого і наступних октетів записується значення довжини блоку закодованої інформації; в перший октет записується кількість додаткових блоків довжини, починаючи з другого; біт 8 першого октету встановлюється в 1. Така форма представлення октетів називається довгою формою (DEFINITE_LONG).

Загальний алгоритм кодування довжин поданий на рисунку 3.12.

First length octet

Form	Bits							
	8	7	6	5	4	3	2	1
Definite, short	0	Length (0–127)						
Indefinite	1	0						
Definite, long	1	Number of following octets (1–126)						
Reserved	1	127						

Рисунок 3.12 — Кодування довжин.

Розроблений протокол використовує лише наступні види довжин ASN.1:

- DEFINITE_SHORT;
- DEFINITE_LONG.

В реалізованому протоколі пусті списки надсилаються за допомогою типу SEQUENCE з довжиною 0.

Протокол використовує надає наступні TCP повідомлення типу “запит клієнта”:

- повідомлення сесії клієнта (створення та знищення);

- повідомлення аналізу первинного тексту;
- повідомлення сесії документу (створення та знищення);
- повідомлення діагностики документу;
- повідомлення розумного доповнення.

Кожне повідомлення типу “запит клієнта” має відповідне повідомлення типу “відповідь сервера”.

Протокол легко масштабується та доповнюється, можлива реалізація приведення протоколу старих версій клієнта до нових версій сервера.

Розмір повідомлень розробленого протоколу в середньому у 15-30 разів менше у порівнянні з JSON RPC протоколом (рисунок 3.13), залежить від формату кодування символів (UTF-8, UTF-16…) та способу кодування даних.

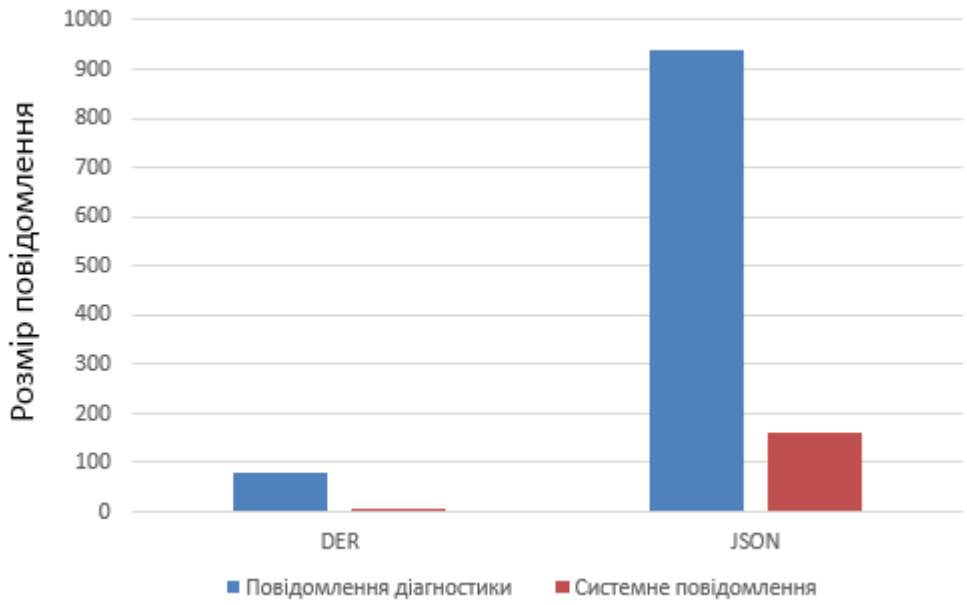

Рисунок 3.13 — Порівняння розробленого протоколу відносно JSON RPC.

3.2.2 Лексичний аналізатор

Лексичний аналіз (розпізнавання вхідної послідовності на масив лексем) є першою фазою аналізу тексту.

Програма, яка здійснює лексичний аналіз, називається лексичним аналізатором або сканером.

Реалізований лексичний аналізатор підтримує наступні типи лексем:

- WHITESPACE;
- NEW_LINE;
- WORD;
- SEPARATOR;
- NUMBER;
- UNKNOWN (у випадку, якщо вхідна лексема містить непідтримуючі символи або не задовільняє правилам лексемотворення).

Для типу SEPARATOR підтримуються наступні символи: '.', '!', '?', ',', '-', ';'.

Лексичний аналізатор підтримує апостроф ('', '\') та римську систему числення. Підтримка римської системи числення відбувається наступним чином: якщо поточна лексема має тимчасовий тип WORD і якщо вона складається лише з відповідних символів римської системи числення ('I', 'V', 'X', 'L', 'C', 'D', 'M'), відповідна лексема змінює свій тип на NUMBER.

Перед проведення лексичного аналізу перший символ тексту перевіряється на збіжність с маркером порядку байтів (Byte-order mark, BOM) та видаляється з аналізу, оскільки в такому випадку не будуть збігатися діапазони значень лексем (на одиницю до першого переносу строки), що призведе до неправильної роботи модулю діагностики клієнта. Після чого сам маркер порядку байтів використовується за призначенням.

Після роботи лексичного аналізатора кожна лексема містить інформацію про: контент, тип, діапазон.

На наступному етапі відбувається розпізнавання речень по спеціальним символам ('.', '!', '?'). Після роботи аналізатора речень на виході отримується об’єкт типу “Document”, який має значення, діапазон та масив речень. Об’єкт типу “Sentence” має значення, діапазон та посилання на документ.

3.2.3 Виконання запитів мовного клієнта

Для надання підтримки одночасної роботи з декількома клієнтами, мовний сервер зберігає по одній обгортці над клієнтом, відповідно до кожного TCP клієнта.

Для надання підтримки одночасної роботи з декількома документами, мовний сервер зберігає в обгортці над клієнтом множину процесорів документів.

Процесор документу зберігає множину поточних операцій від мовного клієнта над документом, операції виконуються послідно за допомогою черги операцій.

Мовний сервер виконує наступні типи операцій:

- BaseOperation — абстрактний базовий клас для операцій;
- HoverOperation — операція забезпечення підказки;
- SuggestionOperation — операція забезпечення розумного доповнення;
- AnalyzeBaseOperation — абстрактний базовий клас для операцій аналізу;
- DiagnosticOperation — операція аналізу поточного тексту;
- AnalyzeOperation — операція аналізу первинного тексту.

Кожна операція зберігає дані про порядковий номер запиту та підтримує можливість її скасування під час виконання.

3.2.4 Забезпечення підтримки природномовних текстів

Реалізований мовний сервер підтримує наступну функціональність:

- доповнення тексту без врахування флексій;
- доповнення тексту врахуванням флексій;
- діагностика тексту (перевірка орфографії).

Для можливості надання даної мовної функціональності, використовується граматичний словник mphdict [30].

mphdict вільно розповсюджується та його реєстр складає 261499 слів (на вересень 2016 року) і є найбільшим словником такого типу. Він має опис внутрішньої структури бази даних словника. mphdict надає вільний доступ до словника словозмінної класифікації. Усі бази даних словника доступні під ліцензією Open Database License і надаються для використання, правок та змін відповідно до потреб користувачів.

Проект містить SQLite бази даних граматичних словників, за основу був взятий "Орфографічний словник української мови" С. Головащука, створений на основі 4-го видання "Українського правопису" (1993).

Для можливості використання словника були додані наступні Nuget-пакети: Microsoft.Data.Sqlite.Core, Microsoft.Data.Sqlite, SQLitePCLRaw.core, SQLitePCLRaw.provider.e_sqlite3.net45.

mphdict вмішає наступні словники:

- граматичний словник (рисунок 3.14);
- словник синонімів;
- етимологічний словник (містить відомості про етимологію слів).

Граматичний словник вміщає наступні основні таблиці:

- nom — таблиця реєстрових одиниць словника;
- indents — таблиця словозмінних класів;
- flexes — таблиця квазіфлексій словозмінних класів;
- parts — таблиця частин мови;
- accents_class — таблиця класів наголосів (акцентуаційних класів);
- gr — таблиця граматичних категорій.

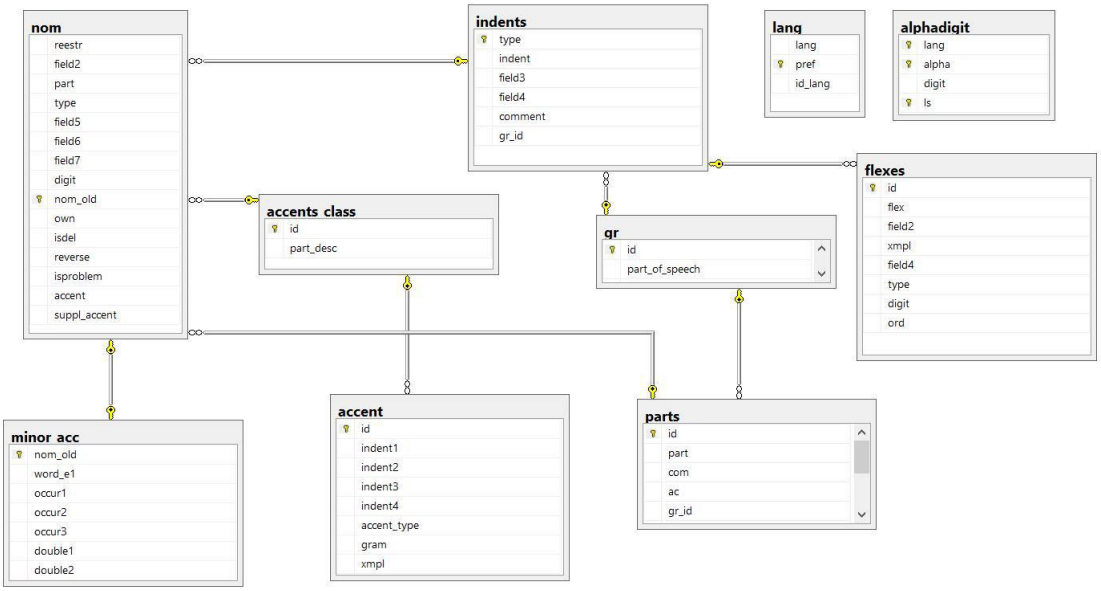

Рисунок 3.14 — Структура бази даних граматичного словника.

В дані роботі використовуються наступні таблиці та відповідні в них колонки: nom (таблиця 3.1), indents (таблиця 3.2), flexes (таблиця 3.3), parts (таблиця 3.4).

Таблиця 3.1.

Name	Type	Comments
reestr	nvarchar(255)	Реєстрове слово
part	smallint	Частина мови (табл. [parts])
type	smallint	Номер словозмінного класу (табл. [indents])

Таблиця 3.2.

Name	Type	Comments
type	smallint	унікальний ідентифікатор запису (номер словозмінного класу)
indent	integer	кількість літер, які потрібно відрізати від кінця слова, щоб залишилась квазіоснова
gr_id	smallint	ідентифікатор гр.кат. з табл. [gr]

Таблиця 3.3.

Name	Type	Comments
id	integer	унікальний ідентифікатор запису
flex	nvarchar(255)	квазіфлексія
field2	integer	номер граматичної категорії
type	smallint	номер словозмінного класу

Таблиця 3.4.

Name	Type	Comments
id	smallint	унікальний ідентифікатор запису, код частини мови
gr_id	smallint	ідентифікатор гр.кат. з табл. [gr]

Name	Type	Comments
rid	integer	рід
mnozh	integer	

Номер граматичної категорії використовуються для знаходження інформації про відмінок, кількість та рід.

Ідентифікатор граматичної категорії використовується для знаходження частини мови.

Для оптимізації аналізу вхідного тексту, а саме пошуку словозмінної форми, було проведено створення спеціального методу у словнику mphdict, який під час першого запуску сервера та доступу до бази, виконує прохід по всім словам та формує словник флексій до цієї словоформи. Це призвело до відмови від використання неоптимізованих алгоритмів пошуку флексій у mphdict, яке включає SQL операції LINQ-to-Entities через Entity Framework та пришвидшило пошук слова приблизно у 30 - 50 раз.

Словник флексій формується наступним чином:

- береться запис з таблиці «nom»;
- визначається основа слова;
- видаляється у слова наголос;
- робиться вибірка записів з таблиці «flexes», що мають словозмінний клас.

В даній роботі використовується лише граматичний словник, тому для прискорення запуску сервера було видалено завантаження інших словників. З лексичного словника використовуються лише таблиці nom, flexes, parts, тому завантаження інших таблиць також було видалено.

Для роботи з базою даних створений адаптер для роботи з даними словника. Під час запуску сервера, адаптер проходить по всьому словнику та зберігає інформацію про слово, при цьому роблячи маршалинг даних, формуючи загальний список слів. Після чого усі слова сортуються по постійній частині слова для

оптимізації пошуку слова. Первинна обробка даних бази займає приблизно 10-20 сек на процесорі i7-5500u.

Перевірка орфографії складається з наступних етапів:

- пошук слів з вхідного тексту у словнику;
- пошук можливих корекцій для невідомих слів;
- додаткова дія для корекції з урахуванням морфології.

Під час викликів аналізаторів тексту для пошуку слова з даних словника, адаптер повертає слово з кешу (між усіма сесіями), а при його відсутності в кеші, з загального списку.

Пошук слова починається з першого слова, яке починається з тієї ж букви, що і вхідне, відбувається прохід по всім лексіям, та закінчується у разі неуспішного пошуку останнім слово, яке починається з тієї ж букви, що і вхідне. При цьому потрібно обробляти спеціальні випадки слів, коли постійна частина слова пуста, наприклад: я-мені, він-йому, вона-їй.

Кожне слово з бази надає наступну інформацію: тип, постійну частину слова, рід, масив флексій, додаткову інформацію, яка залежить від типу.

Тип слова є значенням з наступного списку: NOUN, ADJECTIVE, NUMERATOR, PRONOUN, VERB, PARTICIPLE, GERUND, ADVERB, CONJUNCTION, PREPOSITION, FRACTION.

Додаткова інформація містить інформацію про: тип числівника та тип дієслова для відповідних типів слова.

Теперішня реалізація мовного сервера не підтримує дієслова які мають одночасно доконаний та недоконаний вид — двовидові дієслова, наприклад: мовити, ранити, женити.

Флексія містить інформацію про: закінчення змінних морфем-афіксів, відмінок, число, рід та час.

Пошук можливих корекцій складається з наступних частин:

- механізм селекції;
- модель кандидатів;
- мовна модель P(c);

- модель помилок P(w|c).

Механізм селекції — обрання найбільш ймовірних кандидатів.

Модель кандидатів — множина можливих змін невідомого слова, складається з наступних підмножин, де n — довжина слова:

- множина видалень (n елементів);
- множина вставок (34 * (n + 1) елементів);
- множина переміщень (n - 1 елементів);
- множина заміщень (34 * n елементів).

Усього модель кандидатів складається з 70 * n + 34 елементів. Після створення множини кандидатів відбувається пошук кожного потенційного кандидата у словнику.

Мовна модель P(c) формує ймовірність використання слова “c” у тексті, наприклад, слово “та” зустрічається частіше, ніж слово “там”.

Модель помилок P(w|c) формує ймовірність, що для слова “c” автором помилково було написано слово “w”.

Якщо невідоме слово зустрічається в тексті, що надав користувач для аналізу перед роботою з документами, це слово провайдером діагностики не враховується.

Провайдер діагностики підтримує наступні типи повідомлень діагностики:

- ERROR;
- WARNING.

Провайдер доповнень використовує відсортований по алфавіту масив суфіксних закінчень для кожного слова у реченні, який формується при первинному аналізі тексту. Масив суфіксних закінчень включає лише токени типу “Word” та “Whitespace”.

Після отримання повідомлення від клієнта про запит на отримання закінчень, провайдер доповнень отримує слово, чиї координати вміщають позицію курсору. Після чого за допомогою бінарного пошуку знаходиться підмасив суфіксів.

Якщо результат пустий та виконуються необхідні умови створюються суфікси з використанням додаткових даних. В даній роботі необхідною умовою є використання прикметника та іменника, а додатковими даними є список флексій.

Вихідні дані відсортовуються за частотою використання, відокремлюються перші доповнення, після чого вони посилаються на мовний клієнт.

Висновки до розділу

Мовним клієнтом, який є розширенням для VS Code виконується в окремому процесі спільного хосту, використовує особливості мови TypeScript, використовує події VS Code. Мовний клієнт використовує явну реєстрацію провайдерів у методі активації розширення.

Дане розширення забезпечує високий рівень людино-машинної взаємодії та задовільняє основним критеріям якісного інтерфейсу та зберігає поточні налаштування користувача до глобальних налаштувань.

Мовний сервер обмінюється повідомленнями з мовними клієнтами за допомогою протоколу мовного сервера.

Реалізований мовний сервер підтримує наступну функціональність:

- доповнення тексту без врахування флексій;
- доповнення тексту врахуванням флексій
- діагностика тексту.

4. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

В даний час існують аналогічні рішення для обробки текстів написаних природною мовою, проте вони мають ряд недоліків, не використовують уніфіковані модулі та не мають можливості для їх подальшого розширення.

Тому було би доцільним створити стартап-проект, який надасть можливість розробникам додавати мовну підтримку за допомогою програмного інтерфейсу, використовувати уніфікований мовний протокол.

Було проведено маркетинговий аналіз перспектив реалізації рішення та оцінювання їх можливостей їх ринкового впровадження.

Було виконано наступні задачі:

- аналіз ринкового середовища для стартап-проектів із зазначенням факторів впливу;
- побудова ієрархію факторів із зазначенням сутнісних зв’язків між ними, міри та характеру впливу на стан ринку науково-технічних інноваційних розробок;
- розроблення заходів з комерціалізації стартап-проекту;
- формувати систему складових маркетингової стратегії для стартап-проекту.

4.1 Опис ідеї проекту

Опис ідеї проекту наведений у таблиці 4.1.

Відрізняється від наявних аналогів та замінників підтримкою багатьох мов, зокрема української, наявністю розширень до веб-браузерів та текстових редакторів, швидкодією та глибиною аналізу, можливістю придбати програмний інтерфейс для використання у власних проектах.

Таблиця 4.1.

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Надати можливість повноцінного асистентування при створенні текстового контенту для широкого спектра текстових редакторів	Створення та редагування текстового контенту	Повний аналіз тексту у реальному часі в улюбленому текстовому редакторі або веб-браузері

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від наявних аналогів та замінників) порівняно із пропозиціями конкурентів наведений у таблиці 4.2.

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Технікоекономічні характеристики ідеї	товари/концепції конкурентів			слабка	нейтральна	сильна
		Даний проект	Advego	Grammarly			
1	Економічні	Прибуток з користувачів					Гнучка цінова політика
2	Призначення	Повний аналіз тексту користувача	Перевірка орфографії, граматичних помилок, перевірка на плагіат		Обмеження на довжину тексту	відсутність перевірки на плагіат	Повний аналіз тексту користувача
3	Надійності	Безвідмовна робота					
4	Ергономічні	Підтримка багатьох мов, робота через розширення до текстових редакторів та веб-браузерів	Підтримка певної кількості мов, робота виключно через сайт	Підтримка виключно англійської мови		Прив'язаність до інтерфейсу наявних додатків	Наявність розширень для текстових редакторів та веб-браузерів, підтримка багатьох мов
5	Безпеки	Захист особистої інформації користувачів, конфіденційності документів					

4.2 Технологічний аудит ідеї проекту

Технологічний аудит ідеї проекту наведений у таблиці 4.3.

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Асистентування редагування текстів	Орендувати обчислювальний сервер	є	є
2	Асистентування в реальному часі	Розробити протокол швидкісного обміну інформацією з обчислювальним сервером	є	є
3	Розширення до популярних веб-браузерів	Розробити розширення до веб-браузерів	є	є
4	Розширення до текстових редакторів	Розробити розширення до текстових редакторів	є	є
5	Онлайн-редактор	Розробити веб-сайт сервісу	є	є
6	Система оплати	Інтегрувати наявні системи оплати в інтерфейс сайту	є	є
7	Доступний задокументований програмний інтерфейс	Створення та документування програмного інтерфейсу	є	є
8	Забезпечити захист від злому акаунтів та викрадення текстових даних	Розробити системи захисту від злому акаунтів та викрадення текстових даних	є	є

Висновок: Проект можливо реалізувати з технічної точки зору.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Аналіз попиту наведений у таблиці 4.4.

Таблиця 4.4. Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	Більше 20
2	Загальний обсяг продаж	$10 млрд
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	-
5	Специфічні вимоги до стандартизації та сертифікації	
6	Середня норма рентабельності в галузі (або по ринку), %	7

Висновок: ринок є привабливим для входження за попереднім оцінюванням.

Список потенційних груп клієнтів наведений у таблиці 4.5.

Таблиця 4.5. Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Асистентування при наборі друкованого тексту	Письменники, журналісти	Можливість використовувати продукт у зручному та/ або звичному середовищі	Швидкість та надійність роботи у реальному часі, захищеність оригінального контенту
2	Аналіз наявних текстів	Редактори	Потребують широких налаштувань аналітики	Повноцінний аналіз, захищеність оригінального контенту
3	Програмний інтерфейс	Розробники ПЗ	Потребують гнучкий програмний інтерфейс для популярних технологій розробки ПЗ	Наявність детальної документації

Фактори, що сприяють ринковому впровадженню проекту, та фактори, що йому перешкоджають подані в порядку зменшення значущості в таблиці 4.6, фактори можливостей наведені в таблиці 4.7.

Таблиця 4.6. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Фінансування	Проблема де взяти гроші, особливо на початковій стадії запуску проекту	Пошук інвесторів
2	Популярність сервісу	Низький інтерес до сервісу та недостатньо велика кількість користувачів	Реклама для збільшення аудиторії користувачів
3	Реклама	Можливі складнощі в розробці маркетингової кампанії та укладенні відповідних угод	
4	Недостатньо велика кількість підтримуваних мов	Низький інтерес користувачів до сервісу	Прикласти зусилля для збільшення кількості мов
5	Захист оригінального контенту	Можливі проблеми з довірою до ресурсу через використання можливостей сервісу для викрадення оригінального текстового контенту	Розробити надійну систему захисту
6	Безпека користувачів	Шахраї можуть отримати доступ до акаунтів користувачів, їх персональних даних та коштів	Розробити надійну систему захисту
7	Надійність	Сервіс повинен забезпечувати безвідмовну роботу, проблема пікового навантаження	Орендувати сервер на умовах «платиш тільки за те, що використовуєш і коли використовуєш»
8	Розробка сервісів аналітики	Можливі складнощі при розробці, відставання від термінів тощо	Найняти додатковий персонал, змінити терміни тощо
9	Запуск і супровід	Можливі непередбачені проблеми	Створити команду технічної підтримки

Таблиця 4.7. Фактори можливостей

№	Фактор	Зміст можливостей	Можлива реакція компанії
1	Прибуток	Отримання прибутку	Заходи для збільшення рентабельності
2	Гнучка система оплати	Можливі різні способи оплати	Забезпечити різні способи оплати
3	Широкі можливості аналітики та асистування	Широкий функціонал для користувачів	Забезпечити користувачів необхідним функціоналом
4	Доступний програмний інтерфейс	Розробники можуть придбати програмний інтерфейс для доступу до аналітичного сервера	Забезпечити розробників необхідним функціоналом

Ступеневий аналіз конкуренції на ринку наведений в таблиці 4.8, аналіз конкуренції в галузі за М. Портером наведений в таблиці 4.9.

Таблиця 4.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія 1-го роду	Існує певна кількість схожих сервісів	Забезпечити користувачів ширшими функціональними можливостями
2. За рівнем конкурентної боротьби: глобальний	Схожі продукти доступні в більшості країнах світу для різних мов	Забезпечити доступність в усіх країнах, де це можливо. Забезпечити підтримку якомога більшої кількості мов
3. За галузевою ознакою: внутрішньогалузева	Конкуренція в межах галузі обробки природномовних текстів	Необхідно зробити послуги кращими та зручнішими ніж в конкурентів
4. Конкуренція за видами товарів: товарно-видова	Можлива конкуренція між постачальниками схожих за функціоналом продуктів	Необхідно зробити послуги кращими та зручнішими ніж в конкурентів

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5. За характером конкурентних переваг: цінова та нецінова	Цінова – зробити продукт дешевшим ніж в конкурентів, нецінова – полягає в якості наданих послуг	Цінова – зекономити за рахунок більш оптимізованих алгоритмів обробки. Нецінова – широкі можливості асистування та аналізу текстів
6. За інтенсивністю: не марочна	Конкуренція за надання послуг асистування та аналітики текстів	Розкрутка та набір популярності власного продукту

Таблиця 4.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
	Advego, Grammarly, Microsoft	Бар'єри входження в ринок: - Гнучкі ціни - Розмір капіталовкладень - Зв'язки з рекламодавцями та інфлюенсерами - Широкий спектр можливостей	Фактори сили постачальників: - Змінні витрати постачальників	Фактори сили: - Змінні витрати - Високий рівень чутливості до зміни цін - Контроль якості	Фактори загроз: - Ціна - Лояльність споживачів
Висновки	Висока інтенсивність конкурентної боротьби	Є можливості входу в ринок Є потенційні конкуренти Строки виходу їх на ринок 0.5-1 рік	Провайдери обчислювальних сервісів можуть накладати певні обмеження на трафік та потребувати додаткової оплати	Для клієнтів найбільш важлива цінова доступність послуг, захист персональних даних, якість послуг	Обмеження для роботи на ринку через товари замінники: обмеження економічної рентабельності продукту

Висновок: за умови реалізації гнучкої системи оплати, широких можливостей аналітики та асистування та підтримки багатьох мов проект буде конкурентоспроможним.

Обґрунтування факторів конкурентоспроможності наведено в таблиці 4.10.

Таблиця 4.10. Обґрунтування факторів конкурентоспроможності.

№	Фактор конкуренто спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Гнучка система вибору способу оплати	Можливість оплати різними сервісами, використовувати продукт за різними рівнями підписки, а також різні цінові політики для різних категорій користувачів забезпечують конкурентоспроможність проекту
2	Широкі можливості аналітики та асистування в реальному часі, інтегрованість	Функціонал продукту є ширшим та якіснішим ніж у конкурентів, продукт може бути інтегрований в наявні популярні продукти: веб-браузери та текстові редактори.
3	Підтримка багатьох мов	Підтримка багатьох мов приверне значно більшу кількість клієнтів.

Аналіз сильних та слабких сторін стартап-проекту наведений в таблиці 4.11, SWOT- аналіз стартап-проекту наведений в таблиці 4.12.

Таблиця 4.11. Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкурентоспроможності	Рейтинг у порівнянні з конкурентами
1	Гнучка система вибору способу оплати	+2
2	Широкі можливості аналітики та асистування в реальному часі	+3
3	Підтримка багатьох мов	+3

Загальний рейтинг: +8.

Таблиця 4.12. SWOT- аналіз стартап-проекту

Сильні сторони: ● Висока продуктивність, за рахунок використання мовного сервера. ● Можливість використання широкого функціоналу наявних IDE.	Слабкі сторони: ● Відсутність промислового досвіду в предметній області. ● Відсутність серверного балансеру. ● Неможливість забезпечити повну мовну підтримку.
Можливості: ● Платформа генерування інструментів для підтримки інших мов. ● Використання наявних магазинів цифрової дистрибуції. ● Використання AI для більш кращого аналізу текстів.	Загрози: ● Поява конкурентів. ● Відсутність захисту від атак на сервер.

Альтернативи ринкового впровадження стартап-проекту наведені у таблиці 4.13.

Таблиця 4.13. Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Пришвидшення процесу розробки, підвищення надійності, кількості доступного контенту, реклами тощо	Потребує значного зростання фінансових витрат, отримання значного фінансування на початковому етапі малоймовірне	6-7 місяців
2	Розробка проекту власними силами на початковому етапі	Немає потреби в значному фінансуванні	12-15 місяців

Висновок: було обрано варіант № 2, де отримання ресурсів є більш простим та ймовірним.

4.4. Розроблення ринкової стратегії проекту

Опис цільових груп потенційних споживачів наведено у таблиці 4.14.

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Користувачі	висока	високий	висока	важко
2	Розробники ПЗ	середня	середній	висока	легко

Було обрано цільові групи: користувачі (письменники, журналісти, редактори, студенти) та розробники.

Стратегія охоплення ринку: стратегія диференційованого маркетингу (працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу).

4.4.2 Базова стратегія розвитку.

Таблиця 4.15. Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Стратегія зростання	масовий маркетинг	Нижча вартість продукту	Стратегія лідерства по витратах
2	Стратегія стабілізації	стратегія диференційованого маркетингу	Клієнтам надається більше можливостей порівняно із продуктами конкурентів	Стратегія диференціації

В якості базової стратегії було обрано стратегію диференціації.

Визначення базової стратегії конкурентної поведінки наведено у таблиці 4.16, визначення стратегії позиціонування у таблиці 4.17**.**

Таблиця 4.16. Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати наявних у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	частково	так	Так. Такі як ціна послуг	Стратегія виклику лідера

Таблиця 4.17. Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Цінова доступність	Наступальна стратегія	Нижча ціна за рахунок економії серверного навантаження	Ціна, якість та безпека
2	Широкі можливості аналітики та асистування в реальному часі	Наступальна стратегія	Інтелектуальне асистентування в реальному часі, більш глибокий аналіз	

4.5. Розроблення маркетингової програми стартап-проекту

Визначення ключових переваг концепції потенційного товару наведено у таблиці 4.18.

Трирівнева маркетингова модель товару наведена у таблиці 4.19.

Таблиця 4.18. Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Асистентування та аналітика природномовних текстів	Глибокий аналіз та асистентування в реальному часі	- Глибокий аналіз - Асистентування в реальному часі - Підтримка багатьох мов - Інтегрованість - Доступний програмний інтерфейс
2	Інтегрованість в наявні продукти	Інтегрованість у популярні веб-браузери та текстові редактори	
3	Доступ до мовних сервісів	Доступ до швидкого сервісу мовної аналітики	

Таблиця 4.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Товар – аналітика текстового документа та асистування в реальному часі
2. Товар у реальному виконанні	Властивості: 1. Кількість символів. 2. Мова. 3. Рівень аналізу. 2.Швидкість передачі даних.
3. Товар із підкріпленням	Графічний інтерфейс користувача з локалізацією, захист від викрадення даних, інтегрованість в наявні продукти
За рахунок чого потенційний товар буде захищено від копіювання: програмним шляхом.	

Визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар наведено у таблиці 4.20.

Визначення оптимальної системи збуту наведено у таблиці 4.21.

Таблиця 4.20. Визначення меж встановлення ціни

№	Рівень цін на товари замінники (послуги мовного експерта)	Рівень цін на товари-аналоги (використання наявних сервісів)	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2000-14000 грн	0-1800 грн	всі	0-14000 грн

Таблиця 4.21. Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Індивідуальне споживання товарів та послуг	Збут здійснюється власними силами через магазини цифрової дистрибуції	Одноровневий канал	Продукт - користувач

Розроблення концепції маркетингових комунікацій наведено у таблиці 4.22.

Таблиця 4.22. Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Користувачі потребують можливості використовувати функціонал продукту	Інтернет	Персональний продаж, прямий маркетинг	Зацікавити користувачів	Переконати користувачів, що функціонал їм потрібен для продуктивної роботи із текстовими документами
2	Розробники потребують програмний інтерфейс продукту для використання	Інтернет, бізнес зустрічі	Персональний продаж, прямий маркетинг	Зацікавити розробників придбати програмний інтерфейс	Переконати, що продукт - це вирішення проблем для обробки природномовних текстів в їх проектах

Аналіз ефективності та розрахунок собівартості і ціни продукції наведено у таблиці 4.23, таблиці 4.24, таблиці 4.25 та таблиці 4.26.

Таблиця 4.23. Бюджет витрат за місяць

Бюджет витрат (на місяць)	
1. Матеріальні витрати.	
1.1. Аренда сервера	3000
1.2. Хостинг сайту.	2000
1.3 Реклама	10000
2. Інші змінні витрати.	10000
3. Разом змінні витрати (Σ п.1 + п.2).	28250
4. Заробітна плата адміністративного персоналу	100000
5. Нарахування на зарплату (22% від п. 4.).	22000
6.Зарплата основних виробничих робітників.	100000
7. Нарахування на зарплату (22 % від п. 6.).	22000
9. Комунальні послуги	20000
10. Інші постійні виробничі витрати.	5000
11.Разом постійні витрати (Σпп4-10)	279000
12. Разом повна собівартість (Σ п.3 + п.11)	287290
Всього:	18250

Таблиця 4.24. Бюджет продажів (на місяць)

Бюджет продажів (на місяць)			
Вид товарної продукції	Кількість товарної продукції, шт	ціна, грн	Виручка від реалізації товарної продукції, грн
Підписка	1 000	400,00	400 000,00

Таблиця 4.25. Бюджет про прибутки та збитки підприємства (на місяць)

Бюджет про прибутки та збитки підприємства (на місяць)	
Собівартість реалізованої продукції, грн	270
З податком на прибуток (20 %)	336
Прибуток, грн	100000
Чистий прибуток	60000

Таблиця 4.26. Розрахунок ефективності

	Рік 0	Рік 1	Рік 2	Рік 3
Сума інвестицій (тис. грн.)	3250000	-	-	-
Виручка від реалізації, ₴	-	2880000	4800000	7200000
Витрати на експлуатацію, ₴	-	3228000	3228000	3324000
Амортизаційні відрахування, ₴	-	300000	600000	900000
Ставка дисконту, %	-	18	18	18
Грошові потоки, ₴	-	-48000	2172000	4776000
Дисконтовані грошові потоки, ₴	-	-40677.9661	1559896.581	2906821.048
Дисконтовані вигоди, ₴	-	2694915.254	3878195.921	9558000
Дисконтовані витрати, ₴	-	2735593.22	2318299.339	2023089.021

Точка беззбитковості проекту (ТБ) = 27790 / 400 = 694 од. Чиста теперішня вартість проекту (NVP) = 1176039.663

Термін окупності інвестицій (ТО) = 2.581426717. Коефіцієнт вигід/витрат або коефіцієнт дохідності (BCR) = 2.279377301.

Середня рентабельність – 43% (висока).

Індекс прибутковості (PI) = 1.361858358, PI > 1 отже проект ефективний.

Висновки до розділу

Було проведено аналіз ідеї інтелектуального асистента редактора природномовних текстів сервісу та зроблено висновки:

- чи є можливість ринкової комерціалізації проекту — так;
- чи є перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження, стан конкуренції, конкурентоспроможність проекту — так;
- яку альтернативу (варіант) впровадження доцільно обрати для ринкової реалізації проекту — розробка проекту власними силами на початковому етапі;
- чи є доцільною подальша імплементація проекту — так.

ВИСНОВКИ

Було розглянуто особливості природномовних текстів та інструментальне забезпечення для їх використання.

Створено мовний сервер, мовний клієнт (розширення для VS Code) для написання природномовних текстів на прикладі української мови з використанням найбільшого українського граматичного словника.

Мовний сервер обмінюється повідомленнями з мовними клієнтами за допомогою розробленого протоколу мовного сервера.

Використання реалізованого мовного клієнта значно спрощує час для написання та перевірки текстів українською мовою.

Використання реалізованого мовного сервера та розробленого протоколу розширює можливості для надання мовної підтримки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Karen S. Jones. Natural language processing: a historical review //Cambridge: Computer Laboratory, University of Cambridge, 2001. — P.2

2. D'Ulizia, A., Ferri, F., Grifoni, P. (2011) "A Survey of Grammatical Inference Methods for Natural Language Learning", Artificial Intelligence Review, Vol. 36, No. 1, pp. 1–27

3. Hiroki Arimura; Takeshi Shinohara; Setsuko Otsuki (1994). "Finding Minimal Generalizations for Unions of Pattern Languages and Its Application to Inductive Inference from Positive Data" (PDF). Proc. STACS 11. LNCS. 775. Springer. pp. 649–660.

4. Kishorjit, N., Vidya Raj RK., Nirmal Y., and Sivaji B. (2012) "Manipuri Morpheme Identification", Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP), pages 95–108, COLING 2012, Mumbai, December 2012

5. Ljiljana Dolamic, Jacques Savoy. Stemming Approaches for East European Languages (англ.) // CLEF. — 2007.

6. Lovins, Julie Beth. Development of a Stemming Algorithm // Mechanical Translation and Computational Linguistics. — 1968. — T. 11.

7. Brown, Ralf D. "Transfer-rule induction for example-based translation.". 2001.

8. Мартин Фаулер «Предметно-ориентированные языки программирования» Вильямс, 2011 год.

9. Мартин Фаулер Языковой инструментарий: новая жизнь языков предметной области. — 2005.

10. Mernik - Formal and Practical Aspects of Domain-Specific Languages, 2012.

11. Замаруєва І. В. Модель лінгвістичної бази даних в системах автоматичної обробки природномовної текстової інформації / І. В. Замаруєва, В. Б. Толубко, Л. О.

Литвиненко, О. Ю. Ніколаєвський // Інформатика та математичні методи в моделюванні. - 2013. - Т. 3.

12. Winograd, Terry (1971). Procedures as a Representation for Data in a Computer Program for Understanding Natural Language.

13. Грязнухина, Т.О. Система автоматичного морфологічного аналізу українського наукового тексту / Т.О. Грязнухина, М.В. Нікула // Проблеми українізації комп'ютерів. Матеріали 2-ої Міжнар. конф. — Київ, 1993. — С. 42–46.

14. Roger C. Schank and Robert P. Abelson (1977). Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.

15. Анисимов А. В., Марченко А. А. Система обработки текстов на естественном языке // «Штучний інтелект». — 2002. — № 4.

16. Диковицкий В. В., Шишаев М. Г. Обработка текстов естественного языка в моделях поисковых систем // Сборник научных трудов. — 2010.

17. Olaronke G. Iroju, Janet O. Olaleke, A Systematic Review in Natural Language Processing in Healthcare // I.J. Information Technology and Computer Science. — 2015. — #8. — P. 45

18. Nirenburg Sergei and Raskin Victor. Ontological Semantics, 2001.

19. Goldberg, Yoav (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research 57 (2016) 345–420.

20. Пампуха, І.В. Побудова та використання словника квазізакінчень / І.В. Пампуха, Л.О. Литвиненко, О.Ю. Ніколаєвський та ін. // Збірник наукових праць Військового институту КНУ ім. Тараса Шевченка. — К., 2008. — № 14. — С. 154–158.

21. Extending Visual Studio Code [Електронний ресурс], 2018— Режим доступу: https://code.visualstudio.com/docs/extensions/overview.

22. Yo Code — Extension Generator [Електронний ресурс], 2018— Режим доступу: https://code.visualstudio.com/docs/extensions/yocode.

23. Фисун А. П., Гращенко Л. А. Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человеко-компьютерных систем в информатике и информационной безопасности / А.П.

Фисун. — Деп. в ВИНИТИ 15.10. 2004 г. № 1624 – В 2004. — Орел: Орловский государственный университет, 2004. — 169 с.

24. Fitts, Paul M. (June 1954). "The information capacity of the human motor system in controlling the amplitude of movement". Journal of Experimental Psychology. 47 (6): 381–391

25. "A Common Protocol for Languages". Microsoft. 27 June 2016.

26. Krill, Paul (27 June 2016). "Microsoft-backed Language Server Protocol strives for language, tools interoperability". InfoWorld.

27. Burton S. Kaliski Jr. A Layman's Guide to a Subset of ASN.1, BER, and DER (англ.) // RSA Laboratories: Technical Note. — 1993.

28. Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T X6.90, 07/2002.

29. Lin, Huai-An. “Estimation of the Optimal Performance of ASN.1/BER Transfer Syntax”. ACM Computer Communication Review. July 93.

30. Електронні словники/цифрові лексикографічні системи української мови (серія "Цифрове лексикографічне надбання України") [Електронний ресурс], 2018 — Режим доступу: https://github.com/LinguisticAndInformationSystems/mphdict/wiki.

ДОДАТОК А

Підсистема інтелектуального

редактора природномовних текстів

Апробація

УКР.НТУУ «КПІ».ТЕФ.АПЕПС.ТВ42119_19М

Листів 3

2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVII Міжнародної
науково-практичної конференції
молодих вчених та студентів
м. Київ, 23-26 квітня 2019 року,

ТОМ 2

Київ- 2019

поверхні засобами полікоординатних відображень. 62
РОМАНОВА Д.П., мол. вчений
Керівник - доц., к.т.н. Сидоренко Ю.В.
Реалізація підсистеми моделювання розповсюдження лісової пожежі. 63
АНТОНЮК К.В., мол. вчений гр.
Керівник - доц., к.т.н. Сидоренко Ю.В.
Займенники в українському корпусі проекту Universal Dependencies. 64
ДУДНИК В.Ю., аспірант
Керівник - доц., к.т.н. Статівка Ю.І.
Формування релевантних запитів для збільшення конкурентноспроможності на прикладі графічних систем. 65
ОПЕЙДА Р.А., магістрант гр. ТР-71мн
Керівник - проф., д.т.н. Аушева Н.М.
Модифікація алгоритму політочкових перетворень. 66
ГУМЕНЮК Л.М., магістрант гр. ТВ-61мп
Керівник - доц., к.т.н. Сидоренко Ю.В.
Система оптимізації витрат енергії на підтримку температури в розумному будинку. 67
ВІЛЬДА Д.О., магістрант гр. ТР-81мн
Керівник - доц., к.т.н. Михайлова І.Ю.
Система керування командними проектами на базі Office 365 68
ШКОЛЯР М.В., магістрант гр. ТР-81мн
Керівник - доц., к.т.н. Тихоход В.О.
Визначення об'ємної витрати газу через пальники сушильної печі. 69
САПЕЛЮК Р. В., магістрант гр. АВАУ-11, к.т.н., доц. МАТІКО Г.Ф.Керівник - проф., д.т.н. Матіко Ф.Д.
Система планування дипломного проектування на базі Microsoft Office 365. 70
ЗАВІСТОВСЬКА А.І., магістрант гр. ТМ-81мп
Керівник - доц., к.т.н. Тихоход В.О.
Моделювання порцій на основі ізотропних кривих Без'є. 71
ДОРОЩУК Д.В., магістрант гр. ТР-81мн
Керівник - проф., д.т.н. Аушева Н.М.
Підсистема інтелектуального асистування редактора природномовних текстів. 72
ГОЛЬДИЧ Я.Є., магістрант гр. ТВ-81мп
Керівник - доц., к.т.н. Статівка Ю.І.
Компонент рефакторінгу в інтегрованому середовищі розробки Visual Studio . 73
СТЕПАНЮК А.В., студент гр. ТР-51
Керівник - доц., к.т.н. Тихоход В.О.
Система розпізнавання голосової активності в звуковому сигналі в реальному часі. 74
СКИТЕНКО Р.В., студент гр. ТР-52
Керівник - доц., к.т.н. Статівка Ю.І.
Адаптація акустичної моделі до особливостей звукового сигналу. 75
СЕХІН О.П., студент гр. ТР-52
Керівник - доц., к.т.н. Статівка Ю.І.
Інтеграція динамічної бібліотеки математичних розрахунків з веб-сервісом Node.js. 76
ПІДДУБНЯК А.В., студент гр. ТР-52
Керівник - доц., к.т.н. Демчишин А.А.

УДК 004

Магістрант 5 курсу, гр. ТВ-82мп Гольдич Я.Є.
Доц., к.т.н. Стативка Ю.І.

ПІДСИСТЕМА ІНТЕЛЕКТУАЛЬНОГО АСИСТУВАННЯ РЕДАКТОРА ПРИРОДНОМОВНИХ ТЕКСТІВ

Інтелектуальне асистування у редакторів текстів допомагає значно скоротити час написання, рецензування та редагування текстів.

На даний момент майже для кожної штучної мови існує велика кількість редакторів чи повнофункціональних середовищ розробки, оскільки для штучної мови заздалегідь зазначені граматика, алфавіт та словник, вони є повністю формальними мовами (мають скінченну послідовність символів, які описуються правилами певного виду).

Разом з тим проблема асистування створенню природномовних текстів здебільшого зводиться лише до перевірки — орфографії, синтаксису та, можливо, певних стилістичних показників тексту, а підказки рідко виходять за межі доповнення префіксу до леми чи найчастотнішої словоформи.

З метою дослідження проблем асистування розробці природомовних текстів було прийнято рішення розробки програмного прототипу системи, у складі:

- мовного серверу;
- балансеру;
- мовного клієнту.

Мовний сервер є незалежним відносно клієнтів, які мають різну імплементацію на різних пристроях, та програм-контейнерів клієнта. Мовний сервер також дозволить зменшити навантаження на пристрої клієнта, зможе взаємодіяти з мовним клієнтом за допомогою протоколу мовного сервера.

В якості клієнта можна використати будь-які програмні застосунки починаючи від Add-In для Microsoft Word, Microsft Outlook, які працюють с WordEditor на мові гіпертекстової розмітки HTML, закінчуючи розширеннями для редакторів Visual Studio Code, Notepad чи Atom, що дозволяє реалізовувати повний прикладний інтерфейс відповідної програми.

В якості клієнта для даної імплементації було вирішено створити розширення для Visual Studio Code [1] для написання текстів українською мовою. Розширення складається з наступних модулів:

- модуль розумного доповнення тексту;
- модуль діагностики (помилок);
- модуль підказок.

Усі модулі імплементуються безпосередньо, та вказуються у маніфесті [2] розширення.

Розширення підтримує можливість введення (з меню команд або за допомогою клавіш швидкого доступу) та обрання команд для вибору дії користувача.

Планується створений як розширення VS Code мовний інструментарій використовувати для апробації ідей інтелектуального асистування при роботі з текстами українською мовою.

Перелік посилань:

1. Extending Visual Studio Code [Електронний ресурс]. – Режим доступу : https://code.visualstudio.com/docs/extensions/overview
2. Extension Manifest File - package.json [Електронний ресурс]. – Режим доступу : https://code.visualstudio.com/docs/extensionAPI/extension-manifest